MITSUBISHI

三菱汎用シーケンサ

MELSEC Q series　MELSEC L series

MELSEC-Q/L
構造化プログラミングマニュアル
応用関数編

●安全上のご注意●

(ご使用前に必ずお読みください)

MELSEC-Q/L シリーズシーケンサのご使用に際しては, 各製品に付属しているマニュアルおよび付属マニュアルで紹介している関連マニュアルをよくお読みいただくと共に, 安全に対して十分に注意を払って, 正しい取扱いをしていただくようお願いいたします。

製品に付属しているマニュアルは必要なときに読めるよう大切に保管すると共に, 必ず最終ユーザまでお届けいただくようお願いいたします。

●製品の適用について●

(1) 当社シーケンサをご使用いただくにあたりましては，万一シーケンサに故障・不具合などが発生した場合でも重大な事故にいたらない用途であること，および故障・不具合発生時にはバックアップやフェールセーフ機能が機器外部でシステム的に実施されていることをご使用の条件とさせていただきます。

(2) 当社シーケンサは，一般工業などへの用途を対象とした汎用品として設計・製作されています。したがいまして，以下のような機器・システムなどの特殊用途へのご使用については，当社シーケンサの適用を除外させていただきます。万一使用された場合は当社として当社シーケンサの品質，性能，安全に関る一切の責任（債務不履行責任，瑕疵担保責任，品質保証責任，不法行為責任，製造物責任を含むがそれらに限定されない）を負わないものとさせていただきます。
・各電力会社殿の原子力発電所およびその他発電所向けなどの公共への影響が大きい用途
・鉄道各社殿および官公庁殿など，特別な品質保証体制の構築を当社にご要求になる用途
・航空宇宙，医療，鉄道，燃焼・燃料装置，乗用移動体，有人搬送装置，娯楽機械，安全機械など生命，身体，財産に大きな影響が予測される用途
ただし，上記の用途であっても，具体的に使途を限定すること，特別な品質（一般仕様を超えた品質等）をご要求されないこと等を条件に，当社の判断にて当社シーケンサの適用可とする場合もございますので，詳細につきましては当社窓口へご相談ください。

改 訂 履 歴

※ 取扱説明書番号は，本説明書の裏表紙の左下に記載してあります。

印刷日付	※ 取扱説明書番号	改 訂 内 容
2008 年 4 月	SH(名)-080737-A	初版印刷
2008 年 7 月	SH(名)-080737-B	一部修正 3.7.1 項
2008 年 11 月	SH(名)-080737-C	機種追加 Q00UJCPU，Q00UCPU，Q01UCPU，Q10UDHCPU，Q10UDEHCPU，Q20UDHCPU，Q20UDEHCPU 一部追加 2.1.5 項，5.5.2 項，5.5.3 項 一部修正 本マニュアルで使用する総称・略称
2009 年 4 月	SH(名)-080737-D	機種追加 Q00JCPU，Q00CPU，Q01CPU 一部追加 2.2 節，3.1 節 一部修正 1.2 節，2.1 節，2.1.1 項～ 2.1.12 項→ 2.2.1 項～ 2.2.12 項へ変更，3.1 節→ 3.2 節へ変更，4 章，5 章 一部削除 3.2 節～ 3.7 節，付録 1，付録 2
2009 年 12 月	SH(名)-080737-E	機種追加 L02CPU，L26CPU-BT 一部追加 製品の適用について 一部修正 マニュアルについて，1.1 節，1.2 節，3.1 節，4 章
2010 年 1 月	SH(名)-080737-F	機種追加 Q50UDEHCPU，Q100UDEHCPU 一部修正 1.1 節，1.2 節
2010 年 6 月	SH(名)-080737-G	一部追加 5.1.39 項，5.1.40 項，5.1.41 項，5.1.42 項，5.1.43 項，5.1.44 項，5.1.45 項 一部修正 2.2.1 項
2010 年 7 月	SH(名)-080737-H	一部修正 5.1.30 項，5.1.31 項，5.1.33 項，5.6.1 項
2010 年 11 月	SH(名)-080737-I	一部修正 2.2 節，5.1.13 項，5.1.19 項，5.1.25 項，5.1.29 項，5.1.30 項，5.1.31 項，5.1.32 項，5.1.33 項，5.1.34 項，5.1.36 項，5.1.37 項，5.3.6 項，5.5.1 項，5.5.2 項，5.5.3 項，5.5.4 項，5.6.1 項，5.7.1 項，5.11.1 項，5.11.2 項，5.11.3 項，5.11.4 項，5.12.1 項，5.12.2 項，5.12.3 項，5.12.4 項，

印刷日付	*取扱説明書番号	改　訂　内　容
2011 年 1 月	SH(名)-080737-J	一部追加 2.3 節，6 章 一部修正 1.1 節，2.1 節，2.2.3 項，2.2.4 項，2.2.6 項，5.1.8 項，5.1.10 項，5.1.13 項，5.1.31 項，5.1.32 項，5.1.34 項，5.1.38 項，5.2.1 項，5.3.1 項，5.3.2 項，5.3.3 項，5.3.4 項，5.3.6 項，5.3.7 項，5.4.1 項，5.5.1 項，5.6.1 項，5.8.1 項，5.8.2 項，5.8.3 項，5.8.4 項，5.9.1 項，5.9.2 項，5.10.1 項，5.10.2 項，5.11.1 項，5.11.2 項，5.11.3 項，5.11.4 項，5.12.1 項，5.12.2 項，5.12.3 項，5.12.4 項
2011 年 5 月	SH(名)-080737-K	機種追加 L02CPU-P，L26CPU-PBT 一部修正 1.2 節，2.2.1 項，5.1.14 項，5.1.16 項，5.1.17 項，5.1.18 項，5.1.19 項，5.1.45 項，5.2.1 項，5.3.1 項，5.3.2 項，5.3.4 項，5.3.6 項，5.3.7 項，5.5.1 項，5.5.2 項，5.5.3 項，5.5.4 項，5.6.1 項，5.8.3 項，5.8.4 項，5.11.14 項
2012 年 5 月	SH(名)-080737-L	一部修正 はじめに，マニュアルについて，1.1 節，1.2 節
2012 年 11 月	SH(名)-080737-M	プロセス CPU，二重化 CPU，ユニバーサルモデル QCPU，LCPU の機種追加に伴う全面見直し 機種追加 Q02PHCPU，Q06PHCPU，Q12PHCPU，Q25PHCPU，Q12PRHCPU，Q25PRHCPU，Q03UDVCPU，Q04UDVCPU，Q06UDVCPU，Q13UDVCPU，Q26UDVCPU，L02SCPU，L26CPU
2013 年 1 月	SH(名)-080737-N	機種追加 L06CPU 一部修正 1.2 節
2013 年 4 月	SH(名)-080737-O	機種追加 Q04UDPVCPU，Q06UDPVCPU，Q13UDPVCPU，Q26UDPVCPU，L02SCPU-P，L06CPU-P，L26CPU-P 一部修正 1.2 節

本書によって，工業所有権その他の権利の実施に対する保証，または実施権を許諾するものではありません。また
本書の掲載内容の使用により起因する工業所有権上の諸問題については，当社は一切その責任を負うことができません。

© 2008 MITSUBISHI ELECTRIC CORPORATION

はじめに

このたびは，三菱シーケンサ MELSEC-Q/L シリーズをお買い上げいただきまことにありがとうございました。ご使用前に本マニュアルおよび関連マニュアルをよくお読みいただき，プログラミングの仕様を十分ご理解のうえ，正しくご使用くださるようお願い致します。
本マニュアルで紹介するプログラム例を実際のシステムへ流用する場合は，対象システムにおける制御に問題がないことを十分検証ください。

目　次

安全上のご注意 ····· A - 1
製品の適用について ····· A - 2
改　訂　履　歴 ····· A - 3
はじめに ····· A - 5
目　次 ····· A - 5
マニュアルについて ····· A - 9

1章　概　要　1 - 1 ～ 1 - 6

1.1　本マニュアルの位置付け　1 - 2
1.2　本マニュアルで使用する総称・略称　1 - 5

2章　関数一覧　2 - 1 ～ 2 - 8

2.1　関数一覧表の見方　2 - 2
2.2　関数一覧　2 - 3
2.2.1　型変換関数 ····· 2 - 3
2.2.2　単数値変数関数 ····· 2 - 5
2.2.3　算術演算関数 ····· 2 - 5
2.2.4　ビット型ブール関数 ····· 2 - 5
2.2.5　選択関数 ····· 2 - 5
2.2.6　比較関数 ····· 2 - 6
2.2.7　文字列関数 ····· 2 - 6
2.2.8　時刻データ型関数 ····· 2 - 6
2.2.9　二安定ファンクションブロック ····· 2 - 6
2.2.10　エッジ検出ファンクションブロック ····· 2 - 7
2.2.11　カウンタファンクションブロック ····· 2 - 7
2.2.12　タイマファンクションブロック ····· 2 - 7
2.3　オペレーター一覧　2 - 8
2.3.1　算術演算 ····· 2 - 8
2.3.2　ビット型ブール ····· 2 - 8
2.3.3　比較 ····· 2 - 8

3章　関数の構成　3 - 1 ～ 3 - 4

3.1　関数の構成　3 - 2
3.2　入力ピン数可変関数　3 - 3

4章　関数の見方　4 - 1 ～ 4 - 4

5 章　応用関数　5 - 1 ～ 5 - 234

5.1　型変換関数　5 - 2

5.1.1　ビット型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ·····5 - 2
5.1.2　ビット型→文字列型変換 ·····5 - 5
5.1.3　ビット型→ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型変換 ·····5 - 7
5.1.4　ビット型→時間型変換 ····· 5 - 10
5.1.5　ワード［符号付き］型→ダブルワード［符号付き］型変換 ····· 5 - 12
5.1.6　ダブルワード［符号付き］型→ワード［符号付き］型変換 ····· 5 - 14
5.1.7　ワード［符号付き］型，ダブルワード［符号付き］型→ビット型変換 ····· 5 - 16
5.1.8　ワード［符号付き］型，ダブルワード［符号付き］型→単精度実数型変換 ····· 5 - 19
5.1.9　ワード［符号付き］型，ダブルワード［符号付き］型→倍精度実数型変換 ····· 5 - 22
5.1.10　ワード［符号付き］型，ダブルワード［符号付き］型→文字列型変換 ····· 5 - 25
5.1.11　ワード［符号付き］型，ダブルワード［符号付き］型→ワード［符号なし］/ ビット列［16 ビット］型変換 ····· 5 - 29
5.1.12　ワード［符号付き］型，ダブルワード［符号付き］型→ダブルワード［符号なし］/ ビット列［32 ビット］型変換 ····· 5 - 32
5.1.13　ワード［符号付き］型，ダブルワード［符号付き］型→ BCD 型変換····· 5 - 35
5.1.14　ワード［符号付き］型，ダブルワード［符号付き］型→時間型変換 ····· 5 - 38
5.1.15　単精度実数型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 41
5.1.16　倍精度実数型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 44
5.1.17　単精度実数型→倍精度実数型変換 ····· 5 - 47
5.1.18　倍精度実数型→単精度実数型変換 ····· 5 - 49
5.1.19　単精度実数型→文字列型変換 ····· 5 - 51
5.1.20　ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型→ビット型変換 ····· 5 - 55
5.1.21　ワード［符号なし］/ ビット列［16 ビット］型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 58
5.1.22　ダブルワード［符号なし］/ ビット列［32 ビット］型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 61
5.1.23　ワード［符号なし］/ ビット列［16 ビット］型→ダブルワード［符号なし］/ ビット列［32 ビット］型変換 ····· 5 - 64
5.1.24　ダブルワード［符号なし］/ ビット列［32 ビット］型→ワード［符号なし］/ ビット列［16 ビット］型変換 ····· 5 - 66
5.1.25　ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型→文字列型変換 ····· 5 - 69
5.1.26　ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型→時間型変換 ····· 5 - 72
5.1.27　文字列型→ビット型変換 ····· 5 - 75
5.1.28　文字列型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 77
5.1.29　文字列型→単精度実数型変換 ····· 5 - 80
5.1.30　文字列型→ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型変換 ····· 5 - 84
5.1.31　文字列型→時間型変換 ····· 5 - 88
5.1.32　文字列型→ BCD 型変換····· 5 - 91
5.1.33　BCD 型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 96
5.1.34　BCD 型→文字列型変換 ····· 5 - 100
5.1.35　時間型→ビット型変換 ····· 5 - 103
5.1.36　時間型→ワード［符号付き］型，ダブルワード［符号付き］型変換 ····· 5 - 105
5.1.37　時間型→文字列型変換 ····· 5 - 108

5.1.38 時間型→ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/
ビット列［32 ビット］型変換 ····· 5 - 110
5.1.39 ビット配列→ワード［符号付き］型／ワード［符号なし］/
ビット列［16 ビット］型，ダブルワード［符号付き］型／ダブルワード［符号なし］/
ビット列［32 ビット］型変換 ····· 5 - 113
5.1.40 ワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型，
ダブルワード［符号付き］型／ダブルワード［符号なし］/
ビット列［32 ビット］型→ビット配列変換 ····· 5 - 115
5.1.41 ビット配列のコピー ····· 5 - 117
5.1.42 ワード［符号付き］型データの指定ビット読出し ····· 5 - 119
5.1.43 ワード［符号付き］型データの指定ビット書込み ····· 5 - 121
5.1.44 ワード［符号付き］型データの指定ビットコピー ····· 5 - 123
5.1.45 型変換の不要化 ····· 5 - 125
5.2 単数値変数関数 5 - 127
5.2.1 絶対値 ····· 5 - 127
5.3 算術演算関数 5 - 131
5.3.1 加算 ····· 5 - 131
5.3.2 乗算 ····· 5 - 134
5.3.3 減算 ····· 5 - 137
5.3.4 除算 ····· 5 - 140
5.3.5 剰余 ····· 5 - 143
5.3.6 べき乗 ····· 5 - 146
5.3.7 代入 ····· 5 - 149
5.4 ビット型ブール関数 5 - 152
5.4.1 論理積，論理和，排他的論理和，論理否定 ····· 5 - 152
5.5 選択関数 5 - 157
5.5.1 選択値 ····· 5 - 157
5.5.2 最大値，最小値選択 ····· 5 - 160
5.5.3 上下限リミット制御 ····· 5 - 163
5.5.4 マルチプレクサ ····· 5 - 166
5.6 比較関数 5 - 169
5.6.1 比較 ····· 5 - 169
5.7 文字列関数 5 - 172
5.7.1 文字列の抽出 ····· 5 - 172
5.7.2 文字列の結合 ····· 5 - 175
5.7.3 文字列の挿入 ····· 5 - 177
5.7.4 文字列の削除 ····· 5 - 180
5.7.5 文字列の置換 ····· 5 - 183
5.8 時刻データ型関数 5 - 186
5.8.1 加算 ····· 5 - 186
5.8.2 減算 ····· 5 - 189
5.8.3 乗算 ····· 5 - 192
5.8.4 除算 ····· 5 - 195
5.9 二安定ファンクションブロック 5 - 197
5.9.1 二安定ファンクションブロック（セット優先）····· 5 - 197
5.9.2 二安定ファンクションブロック（リセット優先）····· 5 - 200
5.10 エッジ検出ファンクションブロック 5 - 203
5.10.1 立ち上りエッジ検出 ····· 5 - 203
5.10.2 立ち下りエッジ検出 ····· 5 - 206

5.11 カウンタファンクションブロック 5 - 208
5.11.1 アップカウンタ ····· 5 - 208
5.11.2 ダウンカウンタ ····· 5 - 211
5.11.3 アップダウンカウンタ ····· 5 - 214
5.11.4 カウンタファンクションブロック ····· 5 - 218
5.12 タイマファンクションブロック 5 - 220
5.12.1 パルスタイマ ····· 5 - 220
5.12.2 オンディレータイマ ····· 5 - 223
5.12.3 オフディレータイマ ····· 5 - 226
5.12.4 タイマファンクションブロック ····· 5 - 229

6章 オペレータ 6 - 1 ～ 6 - 18

6.1 算術演算 6 - 2
6.1.1 加算 ····· 6 - 2
6.1.2 乗算 ····· 6 - 4
6.1.3 減算 ····· 6 - 6
6.1.4 除算 ····· 6 - 8
6.1.5 剰余 ····· 6 - 10
6.1.6 べき乗 ····· 6 - 11
6.2 ビット型ブール 6 - 13
6.2.1 論理積，論理和，排他的論理和，論理否定 ····· 6 - 13
6.3 比較演算 6 - 16
6.3.1 比較 ····· 6 - 16

索 引 索引 - 1 ～ 索引 - 4

マニュアルについて

関連マニュアル

本製品に関連するマニュアルには，下記のものがあります。

各マニュアルは，必要に応じて本表を参考にご依頼ください。

(1) 構造化プログラミング

マニュアル名称	マニュアル番号（形名コード）	標準価格
MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編） 構造化プログラムの作成に必要なプログラミング方法，プログラミング言語の種類などについて説明しています。（別売）	SH-080735 (13JC17)	¥3,000
MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編） 構造化プログラムで使用できる，シーケンス命令，基本命令，応用命令などの共通命令に関する仕様，機能について説明しています。（別売）	SH-080736 (13JC18)	¥4,000
MELSEC-Q/L 構造化プログラミングマニュアル（特殊命令編） 構造化プログラムで使用できる，ユニット専用命令，PID 制御命令，内蔵 I/O 機能用命令などの特殊命令に関する仕様，機能について説明しています。（別売）	SH-080738 (13JC20)	¥3,000

(2) GX Works2 の操作

マニュアル名称	マニュアル番号（形名コード）	標準価格
GX Works2 Version 1 オペレーティングマニュアル （共通編） GX Works2 のシステム構成，パラメータ設定，オンライン機能の操作方法など，シンプルプロジェクトと構造化プロジェクトに共通な機能を記載しています。（別売）	SH-080730 (13JV90)	¥4,000
GX Works2 Version 1 オペレーティングマニュアル （構造化プロジェクト編） GX Works2 の構造化プロジェクトでのプログラム作成，モニタなどの操作方法について記載しています。（別売）	SH-080732 (13JV92)	¥3,000
はじめよう GX Works2 （構造化プロジェクト編） はじめて GX Works2 を使用する方のために，構造化プロジェクトでのプログラム作成から編集，モニタなどの基本的な操作方法について説明しています。（別売）	SH-080734 (13JY65)	¥1,500

ポイント

オペレーティングマニュアルは，ソフトウェアパッケージの CD-ROM に PDF ファイルで格納されています。単品でマニュアルを希望される場合は，印刷物を別売で用意していますので上記表のマニュアル番号（形名コード）にてご用命願います。

メ　モ

1 概　要

1.1　本マニュアルの位置付け 1-2
1.2　本マニュアルで使用する総称・略称 1-5

1.1 本マニュアルの位置付け

本マニュアルでは，構造化プログラムの作成で使用する応用関数について説明しています。

目的に応じた参照先マニュアルを下記に示します。

各マニュアルの記載内容，マニュアル番号などについては「関連マニュアル」一覧を参照してください。

(1) GX Works2 の操作

目的		GX Works2 インストール手順書	はじめよう GX Works2		GX Works2 Version 1 オペレーティングマニュアル			
			シンプルプロジェクト編	構造化プロジェクト編	共通編	シンプルプロジェクト編	構造化プロジェクト編	インテリジェント機能ユニット操作編
インストール	動作環境，インストール方法を知りたい	詳細						
	USB ドライバのインストール方法を知りたい				詳細			
GX Works2 の各種操作	GX Works2 のすべての機能が知りたい				概要			
	GX Works2 のプロジェクト種別と使用できる言語が知りたい				概要			
	はじめてシンプルプロジェクトを使用するときの基本の操作と手順を知りたい		詳細					
	はじめて構造化プロジェクトを使用するときの基本の操作と手順を知りたい			詳細				
	プロジェクト種別に関係なく使用できる機能の操作方法を知りたい				詳細			
	プログラミング用の機能と操作方法を知りたい				概要	詳細	詳細	
	インテリジェント機能ユニットのデータ設定方法を知りたい							詳細

(2) プログラミングで使用する言語ごとの操作

各言語のプログラミングで使用する命令の詳細は (3) を参照してください。

目的		はじめよう GX Works2		GX Works2 Version 1 オペレーティングマニュアル	
		シンプル プロジェクト編	構造化 プロジェクト編	シンプル プロジェクト編	構造化 プロジェクト編
シンプル プロジェクト	ラダー				
	SFC	*1			
	ST				
構造化 プロジェクト	ラダー				
	SFC	*1			
	構造化ラダー /FBD				
	ST				

*1: MELSAP3，FX シリーズ用 SFC のみ

(3) 各言語のプログラミングで使用する命令の詳細

目的		MELSEC-Q/L/F 構造化プログラミング マニュアル	MELSEC-Q/L 構造化プログラミング マニュアル			MELSEC-Q/L プログラミング マニュアル	MELSEC-Q/L/QnA プログラミング マニュアル		MELSEC-Q プログラミング／構造化プログラミング マニュアル	使用するユニットのマニュアル
		基礎編	共通命令編	特殊命令編	応用関数編	共通命令編	PID 制御命令編	SFC 編	プロセス制御命令編	–
すべての言語	シーケンサ CPU のエラーコード，特殊リレー・特殊レジスタの内容を知りたい									*1 詳細
ラダーを使用する場合	共通命令の種類と詳細を知りたい					詳細				
	インテリジェント機能ユニット用命令の種類と詳細を知りたい									詳細
	ネットワークユニット用命令の種類と詳細を知りたい									詳細
	PID 制御機能用命令の種類と詳細を知りたい						詳細			
	プロセス制御命令の種類と詳細を知りたい								詳細	
SFC を使用する場合	SFC(MELSAP3)の仕様，機能，命令などの詳細を知りたい							詳細		
構造化ラダー /FBD または ST を使用する場合	構造化プログラミングするための基礎を知りたい	詳細								
	共通命令の種類と詳細を知りたい		詳細							
	インテリジェント機能ユニット用命令の種類と詳細を知りたい			概要						詳細
	ネットワークユニット用命令の種類と詳細を知りたい			概要						詳細
	PID 制御機能用命令の種類と詳細を知りたい			概要			詳細			
	応用関数の種類と詳細を知りたい				詳細					
	プロセス制御命令の種類と詳細を知りたい								詳細	

＊ 1: 使用する CPU ユニットのユーザーズマニュアル（ハードウェア設計・保守点検編）を参照してください。

1.2 本マニュアルで使用する総称・略称

本マニュアルでは，ソフトウェアパッケージ，シーケンサ CPU などを，次の総称・略称で表しています。対象形名の明示が必要なときは，ユニット形名を記載しています。

総称／略称	総称・略称の内容
GX Works2	MELSEC シーケンサソフトウェアパッケージの製品名
ベーシックモデル QCPU	Q00JCPU，Q00CPU，Q01CPU の総称。
ハイパフォーマンスモデル QCPU	Q02CPU，Q02HCPU，Q06HCPU，Q12HCPU，Q25HCPU の総称。
プロセス CPU	Q02PHCPU，Q06PHCPU，Q12PHCPU，Q25PHCPU の総称。
二重化 CPU	Q12PRHCPU，Q25PRHCPU の総称。
ユニバーサルモデル QCPU	Q00UJCPU，Q00UCPU，Q01UCPU，Q02UCPU，Q03UDCPU，Q03UDVCPU，Q03UDECPU，Q04UDHCPU，Q04UDVCPU，Q04UDPVCPU，Q04UDEHCPU，Q06UDHCPU，Q06UDVCPU，Q06UDPVCPU，Q06UDEHCPU，Q10UDHCPU，Q10UDEHCPU，Q13UDHCPU，Q13UDVCPU，Q13UDPVCPU，Q13UDEHCPU，Q20UDHCPU，Q20UDEHCPU，Q26UDHCPU，Q26UDVCPU，Q26UDPVCPU，Q26UDEHCPU，Q50UDEHCPU，Q100UDEHCPU の総称。
QCPU(Q モード)	ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU，ユニバーサルモデル QCPU の総称。
LCPU	L02SCPU，L02SCPU-P，L02CPU，L02CPU-P，L06CPU，L06CPU-P，L26CPU，L26CPU-P，L26CPU-BT，L26CPU-PBT の総称。
CPU ユニット	QCPU(Q モード)，LCPU の総称。
パソコン	Windows® が動作するパーソナルコンピュータの総称。
共通命令	シーケンス命令，基本命令，応用命令，データリンク用命令，マルチ CPU 間専用命令，マルチ CPU 間高速通信専用命令，二重化システム用命令の総称。
特殊命令	ユニット専用命令，PID 制御命令，ソケット通信機能用命令，内蔵 I/O 機能用命令，データロギング機能用命令の総称。
応用関数	IEC61131-3 で定義された関数（ファンクションおよびファンクションブロック）の総称。 （シーケンサの内部では，複数の共通命令を組み合わせて実行される。）

メ　モ

2 関数一覧

2.1 関数一覧表の見方 2-2
2.2 関数一覧 2-3

2.1 関数一覧表の見方

2.2 節の関数一覧表は，次のような形式になっています。

関数名	引数	処理内容	説明 ページ
ADD_E	ⓢ1,ⓢ2,...ⓢn,ⓓ (ピン数可変)	入力値の和（ⓢ1+ⓢ2+・・・+ⓢn）を出力します。	5-131
MUL_E	ⓢ1,ⓢ2,...ⓢn,ⓓ (ピン数可変)	入力値の積（ⓢ1×ⓢ2×・・・×ⓢn）を出力します。	5-134
SUB_E	ⓢ1,ⓢ2,ⓓ	入力値の差（ⓢ1−ⓢ2）を出力します。	5-137
DIV_E	ⓢ1,ⓢ2,ⓓ	入力値の商（ⓢ1÷ⓢ2）を出力します。	5-140
MOD(_E)	ⓢ1,ⓢ2,ⓓ	入力値の余り（ⓢ1÷ⓢ2）を出力します。	5-143
EXPT(_E)	ⓢ1,ⓢ2,ⓓ	入力値のべき乗を出力します。	5-146

① ② ③ ④

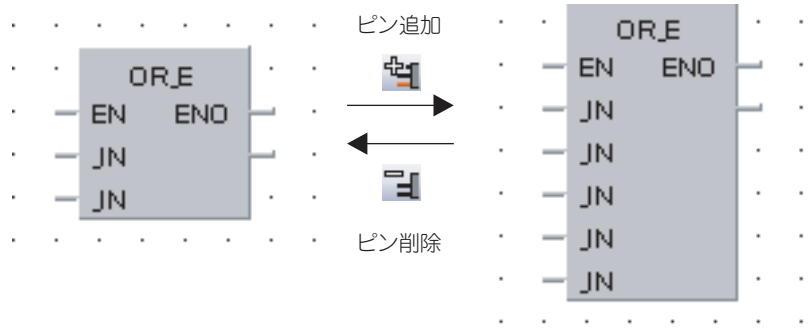
説　明

① ・・・・・・プログラムで使用する関数名を示します。
“関数名 (_E)” は EN/ENO 付き関数として使用します。

② ・・・・・・関数の引数を示します。

ⓢ: ソース・・・・・・・・・・・・・・・・・・・・ 演算前にデータを格納します。
ⓓ: ディスティネーション・・・・・・・ 演算後のデータの行き先を示します。
（ピン数可変）・・・・・・・・・・・・・・・・・ ⓢ（ソース）の数を 2 ～ 28 まで変更できます。

ピン数を変更する場合



③ ・・・・・・各関数の処理内容を示します。

④ ・・・・・・各関数を説明しているページを示します。

2.2 関数一覧

2.2.1 型変換関数

関数名	引数	処理内容	説明ページ
BOOL_TO_INT(_E)	ⓢ,ⓓ	ビット型データをワード［符号付き］型，ダブルワード［符号付き］型データへ変換します。	5-2
BOOL_TO_DINT(_E)	ⓢ,ⓓ		
BOOL_TO_STR(_E)	ⓢ,ⓓ	ビット型データを文字列型データへ変換します。	5-5
BOOL_TO_WORD(_E)	ⓢ,ⓓ	ビット型データをワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型データへ変換します。	5-7
BOOL_TO_DWORD(_E)	ⓢ,ⓓ		
BOOL_TO_TIME(_E)	ⓢ,ⓓ	ビット型データを時間型データへ変換します。	5-10
INT_TO_DINT(_E)	ⓢ,ⓓ	ワード［符号付き］型データをダブルワード［符号付き］型データへ変換します。	5-12
DINT_TO_INT(_E)	ⓢ,ⓓ	ダブルワード［符号付き］型データをワード［符号付き］型データへ変換します。	5-14
INT_TO_BOOL(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データをビット型データへ変換します。	5-16
DINT_TO_BOOL(_E)	ⓢ,ⓓ		
INT_TO_REAL(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データを単精度実数型データへ変換します。	5-19
DINT_TO_REAL(_E)	ⓢ,ⓓ		
INT_TO_LREAL(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データを倍精度実数型データへ変換します。	5-22
DINT_TO_LREAL(_E)	ⓢ,ⓓ		
INT_TO_STR(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データを文字列型データへ変換します。	5-25
DINT_TO_STR(_E)	ⓢ,ⓓ		
INT_TO_WORD(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データをワード［符号なし］/ ビット列［16 ビット］型データへ変換します。	5-29
DINT_TO_WORD(_E)	ⓢ,ⓓ		
INT_TO_DWORD(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データをダブルワード［符号なし］/ ビット列［32 ビット］型データへ変換します。	5-32
DINT_TO_DWORD(_E)	ⓢ,ⓓ		
INT_TO_BCD(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データを BCD 型データへ変換します。	5-35
DINT_TO_BCD(_E)	ⓢ,ⓓ		
INT_TO_TIME(_E)	ⓢ,ⓓ	ワード［符号付き］型，ダブルワード［符号付き］型データを時間型データへ変換します。	5-38
DINT_TO_TIME(_E)	ⓢ,ⓓ		
REAL_TO_INT(_E)	ⓢ,ⓓ	単精度実数型データをワード［符号付き］型，ダブルワード［符号付き］型データへ変換します。	5-41
REAL_TO_DINT(_E)	ⓢ,ⓓ		
LREAL_TO_INT(_E)	ⓢ,ⓓ	倍精度実数型データをワード［符号付き］型，ダブルワード［符号付き］型データへ変換します。	5-44
LREAL_TO_DINT(_E)	ⓢ,ⓓ		
REAL_TO_LREAL(_E)	ⓢ,ⓓ	単精度実数型データを倍精度実数型データへ変換します。	5-47
LREAL_TO_REAL(_E)	ⓢ,ⓓ	倍精度実数型データを単精度実数型データへ変換します。	5-49
REAL_TO_STR(_E)	ⓢ,ⓓ	単精度実数型データを文字列型（指数型式）データへ変換します。	5-51
WORD_TO_BOOL(_E)	ⓢ,ⓓ	ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型データをビット型データへ変換します。	5-55
DWORD_TO_BOOL(_E)	ⓢ,ⓓ		
WORD_TO_INT(_E)	ⓢ,ⓓ	ワード［符号なし］/ ビット列［16 ビット］型データをワード［符号付き］型，ダブルワード［符号付き］型データへ変換します。	5-58
WORD_TO_DINT(_E)	ⓢ,ⓓ		
DWORD_TO_INT(_E)	ⓢ,ⓓ	ダブルワード［符号なし］/ ビット列［32 ビット］型データをワード［符号付き］型，ダブルワード［符号付き］型データへ変換します。	5-61
DWORD_TO_DINT(_E)	ⓢ,ⓓ		
WORD_TO_DWORD(_E)	ⓢ,ⓓ	ワード［符号なし］/ ビット列［16 ビット］型データをダブルワード［符号なし］/ ビット列［32 ビット］型データへ変換します。	5-64

関数名	引数	処理内容	説明ページ
DWORD_TO_WORD(_E)	ⓢ,ⓓ	ダブルワード [符号なし]/ ビット列 [32 ビット] 型データをワード [符号なし]/ ビット列 [16 ビット] 型データへ変換します。	5-66
WORD_TO_STR(_E)	ⓢ,ⓓ	ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型データを文字列型データへ変換します。	5-69
DWORD_TO_STR(_E)	ⓢ,ⓓ		
WORD_TO_TIME(_E)	ⓢ,ⓓ	ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型データを時間型データへ変換します。	5-72
DWORD_TO_TIME(_E)	ⓢ,ⓓ		
STR_TO_BOOL(_E)	ⓢ,ⓓ	文字列型データをビット型データへ変換します。	5-75
STR_TO_INT(_E)	ⓢ,ⓓ	文字列型データをワード [符号付き] 型，ダブルワード [符号付き] 型データへ変換します。	5-77
STR_TO_DINT(_E)	ⓢ,ⓓ		
STR_TO_REAL(_E)	ⓢ,ⓓ	文字列型データを単精度実数型データへ変換します。	5-80
STR_TO_WORD(_E)	ⓢ,ⓓ	文字列型データをワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型データへ変換します。	5-84
STR_TO_DWORD(_E)	ⓢ,ⓓ		
STR_TO_TIME(_E)	ⓢ,ⓓ	文字列型データを時間型データへ変換します。	5-88
STR_TO_BCD(_E)	ⓢ,ⓓ	文字列型データを BCD 型データへ変換します。	5-91
BCD_TO_INT(_E)	ⓢ,ⓓ	BCD 型データをワード [符号付き] 型，ダブルワード [符号付き] 型データへ変換します。	5-96
BCD_TO_DINT(_E)	ⓢ,ⓓ		
BCD_TO_STR(_E)	ⓢ,ⓓ	BCD 型データを文字列型データへ変換します。	5-100
TIME_TO_BOOL(_E)	ⓢ,ⓓ	時間型データをビット型データへ変換します。	5-103
TIME_TO_INT(_E)	ⓢ,ⓓ	時間型データをワード [符号付き] 型，ダブルワード [符号付き] 型データへ変換します。	5-105
TIME_TO_DINT(_E)	ⓢ,ⓓ		
TIME_TO_STR(_E)	ⓢ,ⓓ	時間型データを文字列型データへ変換します。	5-108
TIME_TO_WORD(_E)	ⓢ,ⓓ	時間型データをワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型データへ変換します。	5-110
TIME_TO_DWORD(_E)	ⓢ,ⓓ		
BITARR_TO_INT(_E)	ⓢ,n,ⓓ	ビット配列から指定のビット数をワード [符号付き] 型／ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号付き] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型データへ変換します。	5-113
BITARR_TO_DINT(_E)	ⓢ,n,ⓓ		
INT_TO_BITARR(_E)	ⓢ,n,ⓓ	ワード [符号付き] 型／ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号付き] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型データの下位 n ビットをビット配列に出力します。	5-115
DINT_TO_BITARR(_E)	ⓢ,n,ⓓ		
CPY_BITARR(_E)	ⓢ,n,ⓓ	ビット配列を指定ビットサイズ分コピーします。	5-117
GET_BIT_OF_INT(_E)	ⓢ,n,ⓓ	ワード [符号付き] 型データの指定ビットの値を読み出します。	5-119
SET_BIT_OF_INT(_E)	ⓢ,n,ⓓ	ワード [符号付き] 型データの指定ビットに値を書き込みます。	5-121
CPY_BIT_OF_INT(_E)	ⓢ,n1,n2,ⓓ	ワード [符号付き] 型データの指定ビットを別のワード [符号付き] 型データの指定ビットにコピーします。	5-123
GET_BOOL_ADDR	ⓢ,ⓓ	データの型をビット型に変換します。	5-125
GET_INT_ADDR	ⓢ,ⓓ	データの型をワード [符号付き] 型に変換します。	
GET_WORD_ADDR	ⓢ,ⓓ	データの型をワード [符号なし]/ ビット列 [16 ビット] 型に変換します。	

2.2.2 単数値変数関数

関数名	引数	処理内容	説明ページ
ABS(_E)	ⓢ,ⓓ	入力値の絶対値を出力します。	5-127

2.2.3 算術演算関数

関数名	引数	処理内容	説明ページ
ADD_E	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の和（ⓢ1+ⓢ2+・・・+ⓢn）を出力します。	5-131
MUL_E	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の積（ⓢ1×ⓢ2×・・・×ⓢn）を出力します。	5-134
SUB_E	ⓢ1,ⓢ2,ⓓ	入力値の差（ⓢ1−ⓢ2）を出力します。	5-137
DIV_E	ⓢ1,ⓢ2,ⓓ	入力値の商（ⓢ1÷ⓢ2）を出力します。	5-140
MOD(_E)	ⓢ1,ⓢ2,ⓓ	入力値の余り（ⓢ1÷ⓢ2）を出力します。	5-143
EXPT(_E)	ⓢ1,ⓢ2,ⓓ	入力値のべき乗を出力します。	5-146
MOVE(_E)	ⓢ,ⓓ	入力値をⓓへ代入します。	5-149

2.2.4 ビット型ブール関数

関数名	引数	処理内容	説明ページ
AND_E	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の論理積を出力します。	5-152
OR_E	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の論理和を出力します。	
XOR_E	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の排他的論理和を出力します。	
NOT(_E)	ⓢ,ⓓ	入力値の論理否定を出力します。	

2.2.5 選択関数

関数名	引数	処理内容	説明ページ
SEL(_E)	ⓢ1,ⓢ2,ⓢ3,ⓓ	選択した入力値を出力します。	5-157
MAXIMUM(_E)	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の最大値を出力します。	5-160
MINIMUM(_E)	ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	入力値の最小値を出力します。	
LIMITATION(_E)	ⓢ1,ⓢ2,ⓢ3,ⓓ	上下限リミット制御された入力値を出力します。	5-163
MUX(_E)	n,ⓢ1,ⓢ2,....ⓢn,ⓓ (ピン数可変)	複数入力値の一つを出力します。	5-166

2.2.6 比較関数

関数名	引数	処理内容	説明ページ
GT_E	ⓢ1,ⓢ2,...ⓢn,ⓓ（ピン数可変）	入力値のデータ比較結果を出力します。	5-169
GE_E	ⓢ1,ⓢ2,...ⓢn,ⓓ（ピン数可変）		
EQ_E	ⓢ1,ⓢ2,...ⓢn,ⓓ（ピン数可変）		
LE_E	ⓢ1,ⓢ2,...ⓢn,ⓓ（ピン数可変）		
LT_E	ⓢ1,ⓢ2,...ⓢn,ⓓ（ピン数可変）		
NE_E	ⓢ1,ⓢ2,ⓓ		

2.2.7 文字列関数

関数名	引数	処理内容	説明ページ
MID(_E)	ⓢ,n1,n2,ⓓ	入力された文字列の任意位置から指定文字分を出力します。	5-172
CONCAT(_E)	ⓢ1,ⓢ2,...ⓢn,ⓓ（ピン数可変）	文字列を結合して出力します。	5-175
INSERT(_E)	ⓢ1,ⓢ2,n,ⓓ	文字列の間へ文字列を挿入して出力します。	5-177
DELETE(_E)	ⓢ,n1,n2,ⓓ	文字列の任意の範囲を削除して出力します。	5-180
REPLACE(_E)	ⓢ1,ⓢ2,n1,n2,ⓓ	文字列の任意の範囲を置換して出力します。	5-183

2.2.8 時刻データ型関数

関数名	引数	処理内容	説明ページ
ADD_TIME(_E)	ⓢ1,ⓢ2,ⓓ	入力値（時間型）の和（ⓢ1+ⓢ2）を出力します。	5-186
SUB_TIME(_E)	ⓢ1,ⓢ2,ⓓ	入力値（時間型）の差（ⓢ1−ⓢ2）を出力します。	5-189
MUL_TIME(_E)	ⓢ1,ⓢ2,ⓓ	入力値（時間型）の積（ⓢ1×ⓢ2）を出力します。	5-192
DIV_TIME(_E)	ⓢ1,ⓢ2,ⓓ	入力値（時間型）の商（ⓢ1÷ⓢ2）を出力します。	5-195

2.2.9 二安定ファンクションブロック

関数名	引数	処理内容	説明ページ
SR(_E)	ⓢ1,ⓢ2,ⓓ	2つの入力値を判別し，1（TRUE）または0（FALSE）を出力します。（セット優先）	5-197
RS(_E)	ⓢ1,ⓢ2,ⓓ	2つの入力値を判別し，1（TRUE）または0（FALSE）を出力します。（リセット優先）	5-200

2.2.10 エッジ検出ファンクションブロック

関数名	引数	処理内容	説明ページ
R_TRIG(_E)	ⓢ,ⓓ	信号の立ち上がりを検出してパルス信号を出力します。	5-203
F_TRIG(_E)	ⓢ,ⓓ	信号の立ち下がりを検出してパルス信号を出力します。	5-206

2.2.11 カウンタファンクションブロック

関数名	引数	処理内容	説明ページ
CTU(_E)	ⓢ1,ⓢ2,n,ⓓ1,ⓓ2	信号の立ち上がり回数をカウントアップします。	5-208
CTD(_E)	ⓢ1,ⓢ2,n,ⓓ1,ⓓ2	信号の立ち上がり回数をカウントダウンします。	5-211
CTUD(_E)	ⓢ1,ⓢ2,ⓢ3,ⓢ4, n,ⓓ1,ⓓ2,ⓓ3	信号の立ち上がり回数をカウントアップ／カウントダウンします。	5-214
COUNTER_FB_M	ⓢ1,ⓢ2,ⓢ3,ⓓ1,ⓓ2	信号の立ち上がり回数をⓢ3からⓢ2までカウントアップします。	5-218

2.2.12 タイマファンクションブロック

関数名	引数	処理内容	説明ページ
TP(_E) TP_HIGH(_E)	ⓢ,n,ⓓ1,ⓓ2	指定された時間の間，信号を ON します。	5-220
TON(_E) TON_HIGH(_E)	ⓢ,n,ⓓ1,ⓓ2	指定された時間後に信号を ON します。	5-223
TOF(_E) TOF_HIGH(_E)	ⓢ,n,ⓓ1,ⓓ2	指定された時間後に信号を OFF します。	5-226
TIMER_10_FB_M TIMER_100_FB_M TIMER_HIGH_FB_M TIMER_LOW_FB_M TIMER_CONT_FB_M TIMER_CONTHFB_M	ⓢ1,ⓢ2,ⓢ3,ⓓ1,ⓓ2	入力値ⓢ3からⓢ2で指定された時間後に信号を ON します。	5-229

ポイント

応用関数のファンクション，ファンクションブロックは，複数のシーケンス命令を組み合わせて演算を行います。そのため，応用関数の演算中に割込みが発生すると，意図しない演算結果になる場合があります。
割込みプログラムを使用する場合は，必要に応じて割込禁止／許可命令（DI/EI 命令）を使用してください。

2.3 オペレーター覧

2.3.1 算術演算

オペレータ名		引数	処理内容	説明ページ
構造化ラダー /FBD	ST			
ADD	+	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)	入力値の和（ⓢ1+ⓢ2+・・・+ⓢ26）を出力します。	6-2
MUL	*	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)	入力値の積（ⓢ1×ⓢ2×・・・×ⓢ26）を出力します。	6-4
SUB	−	ⓢ1,ⓢ2,ⓓ	入力値の差（ⓢ1−ⓢ2）を出力します。	6-6
DIV	/	ⓢ1,ⓢ2,ⓓ	入力値の商（ⓢ1÷ⓢ2）を出力します。	6-8
（未対応）	MOD	ⓢ1,ⓢ2,ⓓ	入力値の余り（ⓢ1÷ⓢ2）を出力します。	6-10
（未対応）	**	ⓢ1,ⓢ2,ⓓ	入力値のべき乗を出力します。	6-11

2.3.2 ビット型ブール

オペレータ名		引数	処理内容	説明ページ
構造化ラダー /FBD	ST			
AND	& AND	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)	入力値の論理積を出力します。	6-13
OR	OR	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)	入力値の論理和を出力します。	
XOR	XOR	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)	入力値の排他的論理和を出力します。	
（未対応）	NOT	ⓢ,ⓓ	入力値の論理否定を出力します。	

2.3.3 比較

オペレータ名		引数	処理内容	説明ページ
構造化ラダー /FBD	ST			
GT	>	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)	入力値のデータ比較結果を出力します。	6-16
GE	>=	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)		
EQ	=	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)		
LE	<=	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)		
LT	<	ⓢ1,ⓢ2,....ⓢ26,ⓓ (ピン数可変)		
NE	<>	ⓢ1,ⓢ2,ⓓ		

3 関数の構成

3.1 関数の構成 3-2
3.2 入力ピン数可変関数 3-3

3.1 関数の構成

CPU ユニットで使用できる命令は，関数名と引数に分けることができます。

関数名と引数の用途は，次のようになっています。

- 関数名 ····· その関数の機能を示す。
- 引数 ····· 関数で使用する入出力データを示す。

引数には，ソースデータ，ディスティネーションデータ，実行条件，実行結果があります。

(1) ソースⓢ

(a) ソースは演算で使用するデータです。

(b) 関数で指定するデバイスにより，次のようになります。

- 定数 ····· 演算で使用する数値を指定します。
  プログラム作成時に設定するため，プログラム実行中での変更はできません。
  定数を可変データで使用する場合は，インデックス修飾してください。
- ビットデバイス，ワードデバイス ···· 演算で使用するデータの格納されているデバイスを指定します。
  演算を実行するまでに指定のデバイスへデータを格納しておく必要があります。
  プログラム実行中，指定されたデバイスに格納するデータを変更することにより，その関数で使用するデータを変更できます。

(c) ビットデバイスを使用するソースに，接点を直接入力することはできません。

(2) ディスティネーションⓓ

(a) ディスティネーションには，演算後のデータが格納されます。

(b) ディスティネーションには必ずデータを格納するためのデバイスを設定します。

(c) ビットデバイスを格納するディスティネーションに，コイルを直接接続することはできません。

(3) 実行条件（EN）

(a) 入力変数 EN は関数の実行条件を入力します。

(4) 実行結果（ENO）

(a) 出力変数 ENO は実行結果を出力します。

ポイント

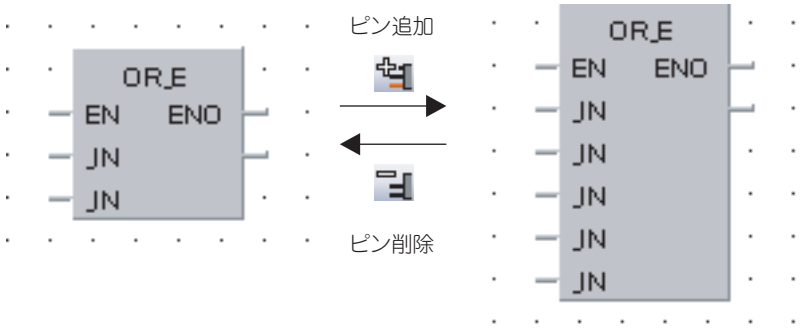
ラベルや構造体などの関数の構成の詳細は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

3.2 入力ピン数可変関数

関数の中には，入力ピン数を変更できるものがあります。

入力ピン数の変更は，対象となる関数を選択して入力ピン数の変更を行います。

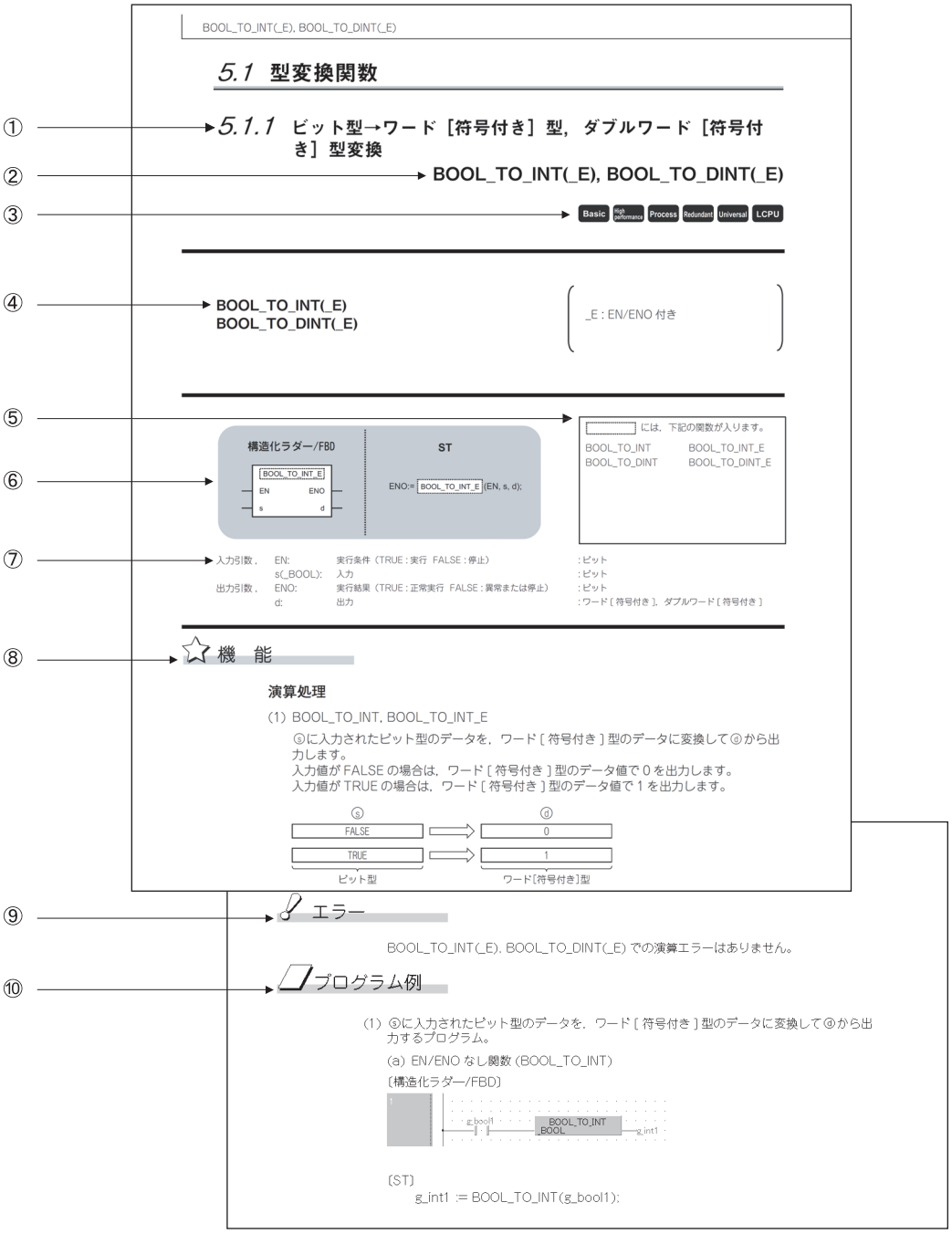
入力ピン数の変更操作は☞GX Works2 Version 1 オペレーティングマニュアル（構造化プロジェクト編）を参照してください。



メ　モ

4 関数の見方

本マニュアルの５章以降では，次のような構成で各関数の説明を行っています。



① 項番号，関数の概要を示しています。

② 説明対象の関数を示しています。

③ 関数を使用できる CPU ユニットを示しています。

アイコン						内　容
ベーシックモデル QCPU	ハイパフォーマンスモデル QCPU	プロセス CPU	二重化 CPU	ユニバーサルモデル QCPU	LCPU	
						通常のアイコンは，該当する関数が使用できることを示します。
						Ver. マーク付きアイコンは，一部制約付きで使用できることを示します。（機能バージョン，ソフトウェアバージョン）
						× マーク付きアイコンは，該当する関数が使用できないことを示します。

④ 関数名を示しています。

⑤ 記述できる関数名を示しています。

⑥ 構造化ラダ ―/FBD，ST での関数の記述形式を示しています。

⑦ 関数の入力引数，出力引数の名称，各引数のデータ型を示しています。
各データ型の詳細は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

⑧ 関数の果たす機能について示しています。

⑨ エラーの有無を示しています。エラーがある場合は，エラーの起きる条件について示しています。

⑩ プログラム例を構造化ラダ ―/FBD，ST について示しています。

メ　モ

5 応用関数

5.1 型変換関数 5-2
5.2 単数値変数関数 5-127
5.3 算術演算関数 5-131
5.4 ビット型ブール関数 5-152
5.5 選択関数 5-157
5.6 比較関数 5-169
5.7 文字列関数 5-172
5.8 時刻データ型関数 5-186
5.9 二安定ファンクションブロック 5-197
5.10 エッジ検出ファンクションブロック 5-203
5.11 カウンタファンクションブロック 5-208
5.12 タイマファンクションブロック 5-220

5.1 型変換関数

5.1.1 ビット型→ワード［符号付き］型，ダブルワード［符号付き］型変換

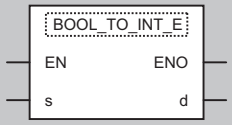
BOOL_TO_INT(_E), BOOL_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

BOOL_TO_INT(_E)
BOOL_TO_DINT(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD



ST

```
ENO:= BOOL_TO_INT_E (EN, s, d);
```

[　　] には，下記の関数が入ります。

BOOL_TO_INT	BOOL_TO_INT_E
BOOL_TO_DINT	BOOL_TO_DINT_E

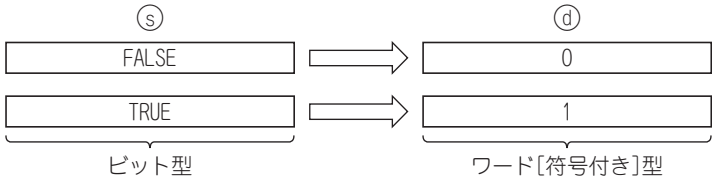
入力引数，	EN:	実行条件（TRUE : 実行　FALSE : 停止）	: ビット
	s(_BOOL):	入力	: ビット
出力引数，	ENO:	実行結果（TRUE : 正常実行　FALSE : 異常または停止）	: ビット
	d:	出力	: ワード［符号付き］，ダブルワード［符号付き］

☆ 機　能

演算処理

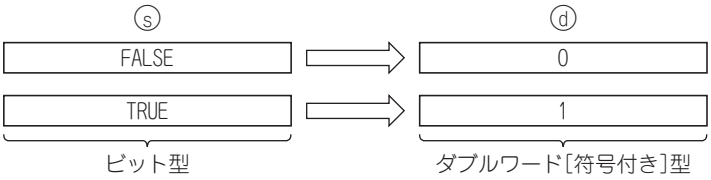
(1) BOOL_TO_INT, BOOL_TO_INT_E

ⓢに入力されたビット型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。
入力値が FALSE の場合は，ワード［符号付き］型のデータ値で 0 を出力します。
入力値が TRUE の場合は，ワード［符号付き］型のデータ値で 1 を出力します。



(2) BOOL_TO_DINT, BOOL_TO_DINT_E

ⓢに入力されたビット型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。
入力値が FALSE の場合は，ダブルワード [符号付き] 型のデータ値で 0 を出力します。
入力値が TRUE の場合は，ダブルワード [符号付き] 型のデータ値で 1 を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

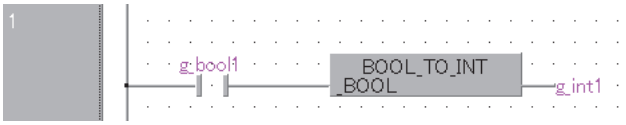
BOOL_TO_INT(_E), BOOL_TO_DINT(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたビット型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BOOL_TO_INT)

〔構造化ラダー/FBD〕

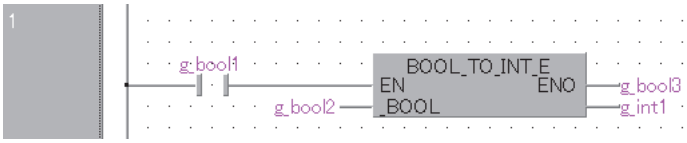


〔ST〕

```
g_int1 := BOOL_TO_INT(g_bool1);
```

(b) EN/ENO 付き関数 (BOOL_TO_INT_E)

〔構造化ラダー/FBD〕



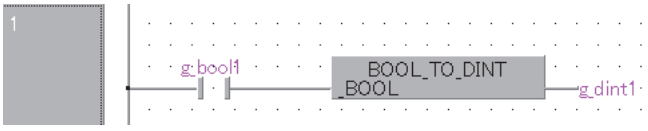
〔ST〕

```
g_bool3 := BOOL_TO_INT_E(g_bool1, g_bool2, g_int1);
```

(2) ⓢに入力されたビット型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BOOL_TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint1 := BOOL_TO_DINT(g_bool1);
```

5.1.2 ビット型→文字列型変換

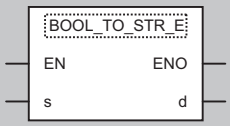
BOOL_TO_STR(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

BOOL_TO_STR(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= BOOL_TO_STR_E(EN, s, d);

[ ] には，下記の関数が入ります。

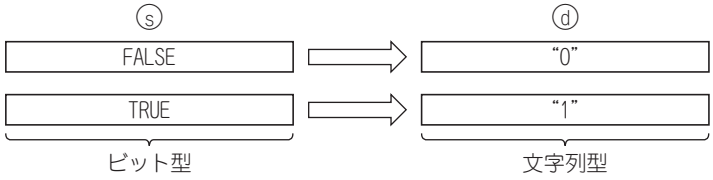
BOOL_TO_STR　　BOOL_TO_STR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_BOOL):	入力	:ビット
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d:	出力	:文字列

機　能

演算処理

ⓢに入力されたビット型のデータを，文字列型のデータに変換してⓓから出力します。
入力値が FALSE の場合は，文字列型のデータ値で "0" を出力します。
入力値が TRUE の場合は，文字列型のデータ値で "1" を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

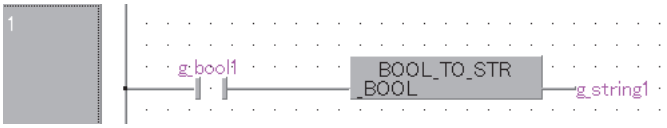
BOOL_TO_STR(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BOOL_TO_STR)

〔構造化ラダー/FBD〕



〔ST〕

```
g_string1 := BOOL_TO_STR (g_bool1);
```

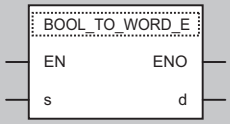
5.1.3 ビット型→ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型変換 BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

BOOL_TO_WORD(_E)
BOOL_TO_DWORD(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= BOOL_TO_WORD_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

BOOL_TO_WORD	BOOL_TO_WORD_E
BOOL_TO_DWORD	BOOL_TO_DWORD_E

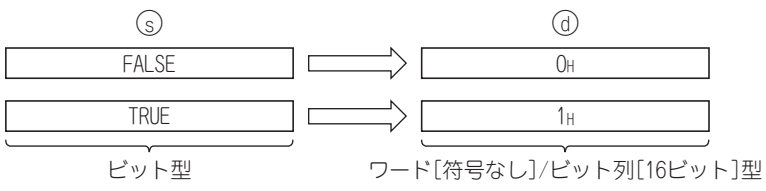
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_BOOL):	入力	: ビット
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］

機　能

演算処理

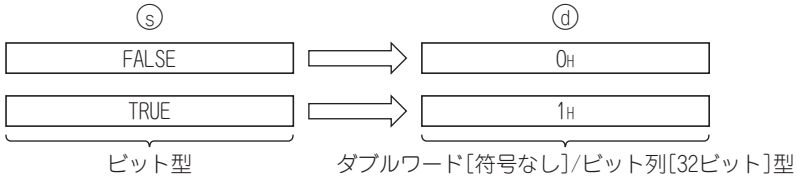
(1) BOOL_TO_WORD, BOOL_TO_WORD_E

ⓢに入力されたビット型のデータを，ワード［符号なし］/ ビット列［16 ビット］型のデータに変換してⓓから出力します。
入力値が FALSE の場合は，ワード［符号なし］/ ビット列［16 ビット］型のデータ値で 0H を出力します。
入力値が TRUE の場合は，ワード［符号なし］/ ビット列［16 ビット］型のデータ値で 1H を出力します。



(2) BOOL_TO_DWORD, BOOL_TO_DWORD_E

ⓢに入力されたビット型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力します。
入力値が FALSE の場合は，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータ値で 0H を出力します。
入力値が TRUE の場合は，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータ値で 1H を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

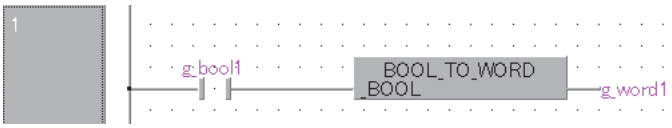
BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたビット型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BOOL_TO_WORD)

〔構造化ラダー/FBD〕

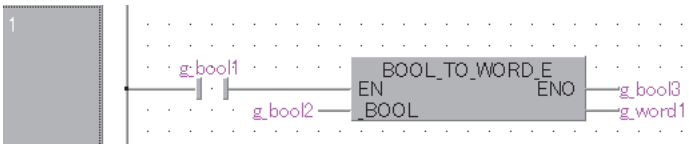


〔ST〕

```
g_word1  := BOOL_TO_WORD (g_bool1);
```

(b) EN/ENO 付き関数 (BOOL_TO_WORD_E)

〔構造化ラダー/FBD〕



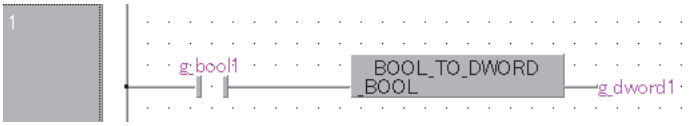
〔ST〕

```
g_bool3 := BOOL_TO_WORD_E (g_bool1, g_bool2, g_word1);
```

(2) ⓢに入力されたビット型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BOOL_TO_DWORD)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dword1 := BOOL_TO_DWORD (g_bool1);
```

5.1.4 ビット型→時間型変換

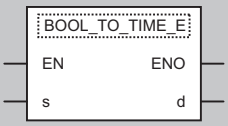
BOOL_TO_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

BOOL_TO_TIME(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= BOOL_TO_TIME_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

BOOL_TO_TIME　　BOOL_TO_TIME_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_BOOL):	入力	:ビット
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d:	出力	:時間

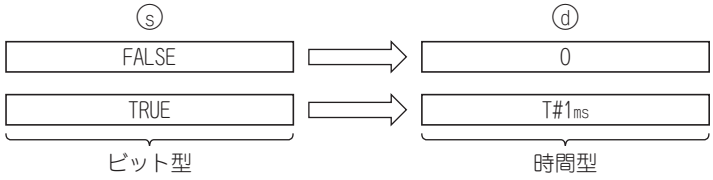
機　能

演算処理

ⓢに入力されたビット型のデータを，時間型のデータに変換してⓓから出力します。

入力値が FALSE の場合は，時間型のデータ値で 0 を出力します。

入力値が TRUE の場合は，時間型のデータ値で 1 を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

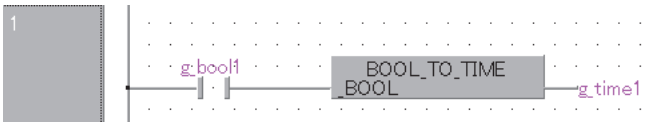
BOOL_TO_TIME(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータを，時間型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BOOL_TO_TIME)

〔構造化ラダー/FBD〕

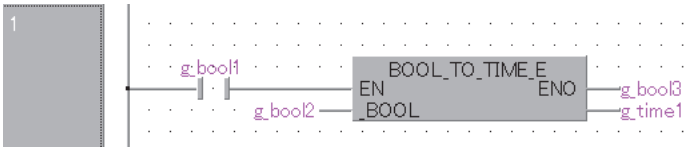


〔ST〕

```
g_time1 := BOOL_TO_TIME (g_bool1);
```

(b) EN/ENO 付き関数 (BOOL_TO_TIME_E)

〔構造化ラダー/FBD〕



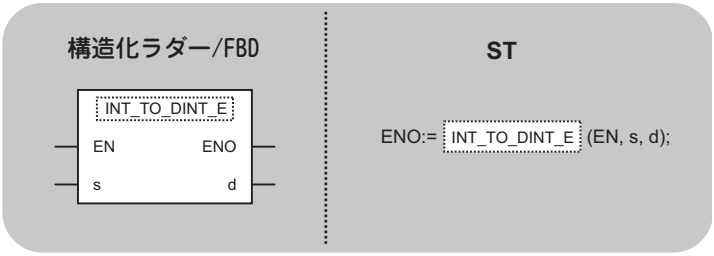
〔ST〕

```
g_bool3 := BOOL_TO_TIME_E (g_bool1, g_bool2, g_time1);
```

5.1.5 ワード［符号付き］型→ダブルワード［符号付き］型変換 INT_TO_DINT(_E)

Basic High performance Process Redundant Universal LCPU

INT_TO_DINT(_E)　（_E : EN/ENO 付き）



[　　] には，下記の関数が入ります。

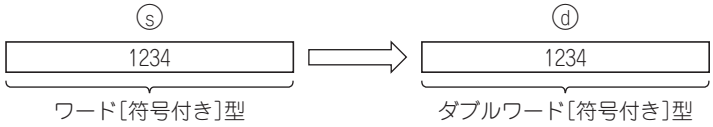
INT_TO_DINT	INT_TO_DINT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_INT):	入力	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ダブルワード［符号付き］

機　能

演算処理

ⓢに入力されたワード［符号付き］型のデータを，ダブルワード［符号付き］型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

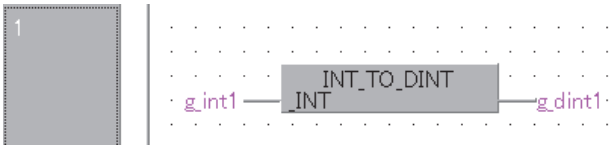
INT_TO_DINT(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたワード [符号付き] 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_DINT)

〔構造化ラダー/FBD〕

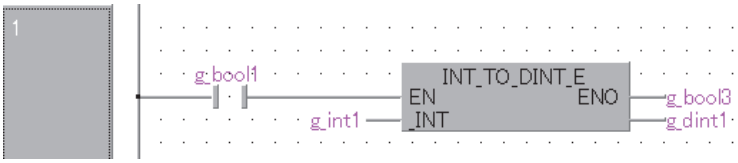


〔ST〕

```
g_dint1 := INT_TO_DINT (g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_DINT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3  := INT_TO_DINT_E (g_bool1, g_int1, g_dint1);
```

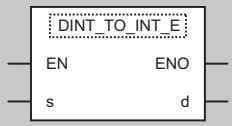
5.1.6 ダブルワード［符号付き］型→ワード［符号付き］型変換 DINT_TO_INT(_E)

Basic High performance Process Redundant Universal LCPU

DINT_TO_INT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= DINT_TO_INT_E (EN, s, d);

[ ] には，下記の関数が入ります。

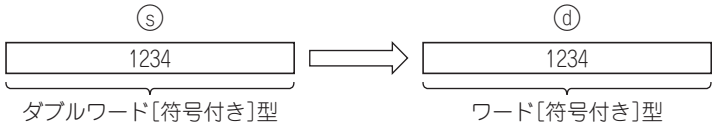
DINT_TO_INT	DINT_TO_INT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_DINT):	入力	:ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ワード［符号付き］

機　能

演算処理

ⓢに入力されたダブルワード［符号付き］型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

ポイント

DINT_TO_INT(_E) を実行時，ⓢから入力されたダブルワード [符号付き] 型データ値の下位 16 ビットの値のみワード [符号付き] 型のデータに変換されます。上位 16 ビットの情報は切り捨てられます。

エラー

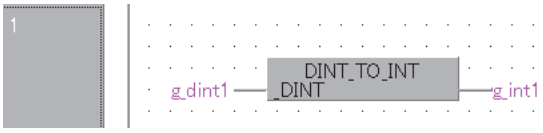
DINT_TO_INT(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたダブルワード [符号付き] 型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_INT)

〔構造化ラダー/FBD〕

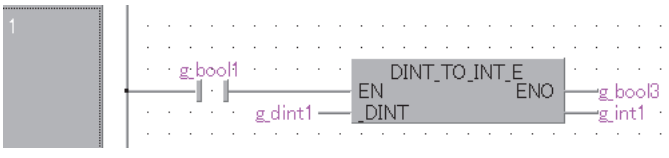


〔ST〕

```
g_int1 := DINT_TO_INT(g_dint1);
```

(b) EN/ENO 付き関数 (DINT_TO_INT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3  := DINT_TO_INT_E (g_bool1, g_dint1, g_int1);
```

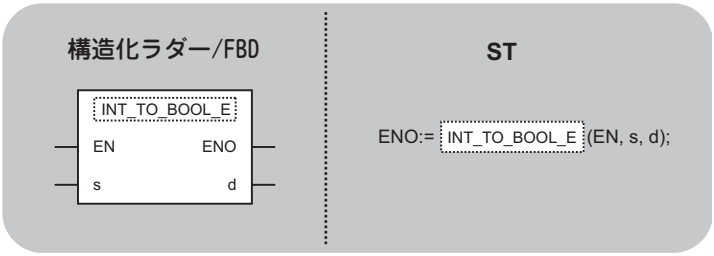
5.1.7 ワード［符号付き］型，ダブルワード［符号付き］型→ビット型変換

INT_TO_BOOL(_E), DINT_TO_BOOL(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_BOOL(_E)
DINT_TO_BOOL(_E)

（_E : EN/ENO 付き）



INT_TO_BOOL_E には，下記の関数が入ります。

INT_TO_BOOL	INT_TO_BOOL_E
DINT_TO_BOOL	DINT_TO_BOOL_E

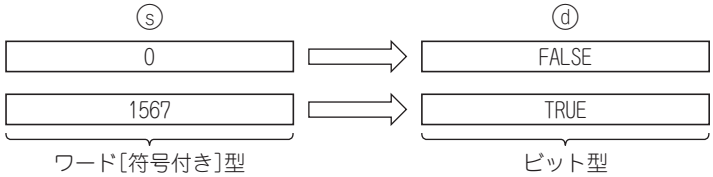
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s(_INT, _DINT):	入力	：ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	：ビット
	d:	出力	：ビット

機　能

演算処理

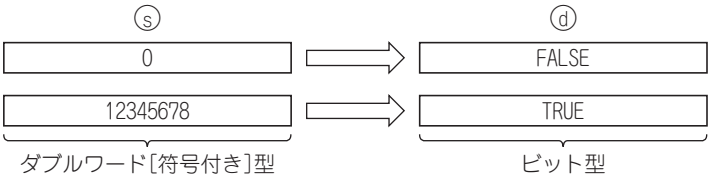
(1) INT_TO_BOOL, INT_TO_BOOL_E

ⓢに入力されたワード［符号付き］型のデータを，ビット型のデータに変換してⓓから出力します。
入力値が 0 の場合は，FALSE を出力します。
入力値が 0 以外の場合は，TRUE を出力します。



(2) DINT_TO_BOOL, DINT_TO_BOOL_E

ⓢに入力されたダブルワード [符号付き] 型のデータを，ビット型のデータに変換してⓓから出力します。
入力値が 0 の場合は，FALSE を出力します。
入力値が 0 以外の場合は，TRUE を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

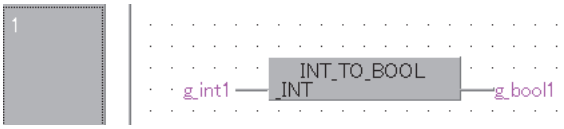
INT_TO_BOOL(_E), DINT_TO_BOOL(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，ビット型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_BOOL)

〔構造化ラダー/FBD〕

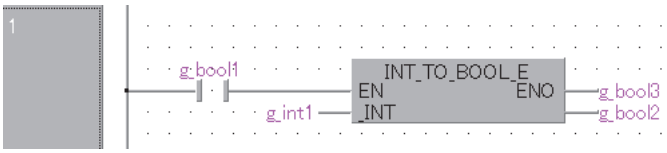


〔ST〕

```
g_bool1 := INT_TO_BOOL(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_BOOL_E)

〔構造化ラダー/FBD〕



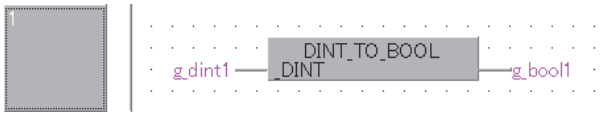
〔ST〕

```
g_bool3 := INT_TO_BOOL_E (g_bool1, g_int1, g_bool2);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，ビット型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_BOOL)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool1 := DINT_TO_BOOL(g_dint1);
```

5.1.8 ワード［符号付き］型，ダブルワード［符号付き］型→単精度実数型変換

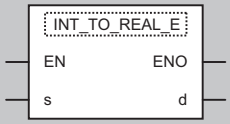
INT_TO_REAL(_E), DINT_TO_REAL(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_REAL(_E)
DINT_TO_REAL(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD



ST

```
ENO:= INT_TO_REAL_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

INT_TO_REAL	INT_TO_REAL_E
DINT_TO_REAL	DINT_TO_REAL_E

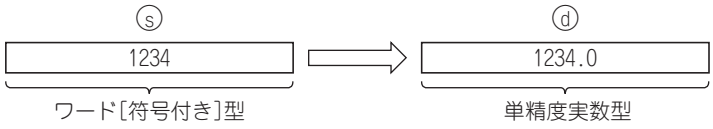
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_INT, _DINT):	入力	: ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 単精度実数

機　能

演算処理

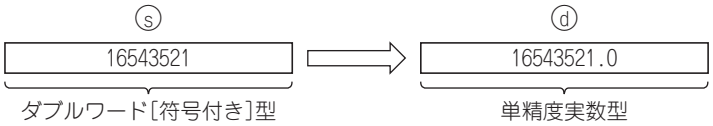
(1) INT_TO_REAL, INT_TO_REAL_E

ⓢに入力されたワード［符号付き］型のデータを，単精度実数型のデータに変換してⓓから出力します。



(2) DINT_TO_REAL, DINT_TO_REAL_E

(a) ⓢに入力されたダブルワード [符号付き] 型のデータを，単精度実数型のデータに変換してⓓから出力します。



(b) 単精度実数型データは，32 ビットの単精度で処理するため，有効桁数は約 7 桁となります。したがって，整数値が –16777216 ～ 16777215 の範囲を超えた場合，変換した値に誤差が生じます。（丸め誤差）

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

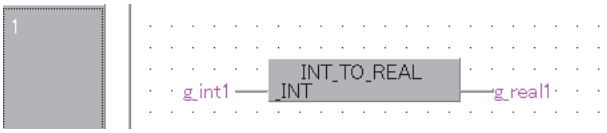
INT_TO_REAL(_E), DINT_TO_REAL(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，単精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_REAL)

〔構造化ラダー/FBD〕

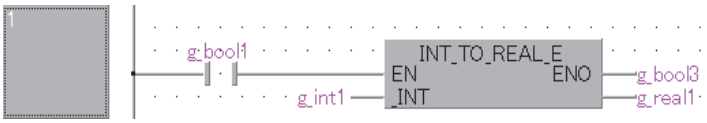


〔ST〕

```
g_real1 := INT_TO_REAL(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_REAL_E)

〔構造化ラダー/FBD〕



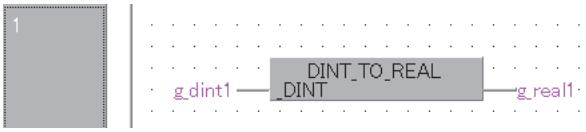
〔ST〕

```
g_bool3 := INT_TO_REAL_E(g_bool1, g_int1, g_real1);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，単精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_REAL)

〔構造化ラダー/FBD〕



〔ST〕

```
g_real1 := DINT_TO_REAL(g_dint1);
```

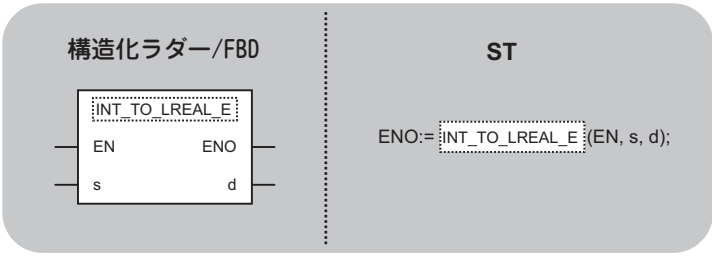
5.1.9 ワード［符号付き］型，ダブルワード［符号付き］型→倍精度実数型変換

INT_TO_LREAL(_E), DINT_TO_LREAL(_E)

Basic (×) High performance (×) Process (×) Redundant (×) Universal LCPU

INT_TO_LREAL(_E)
DINT_TO_LREAL(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

INT_TO_LREAL	INT_TO_LREAL_E
DINT_TO_LREAL	DINT_TO_LREAL_E

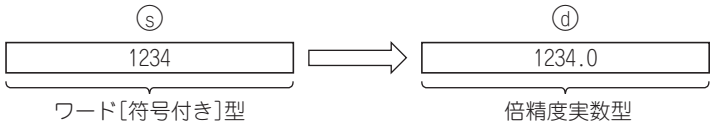
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s(_INT, _DINT):	入力	：ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	：ビット
	d:	出力	：倍精度実数

機　能

演算処理

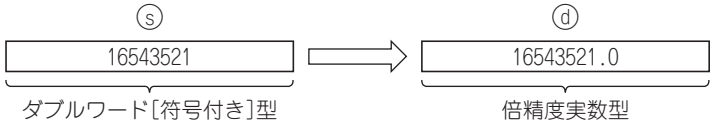
(1) INT_TO_LREAL, INT_TO_LREAL_E

ⓢに入力されたワード［符号付き］型のデータを，倍精度実数型のデータに変換してⓓから出力します。



(2) DINT_TO_LREAL, DINT_TO_LREAL_E

ⓢに入力されたダブルワード［符号付き］型のデータを，倍精度実数型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算出力値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

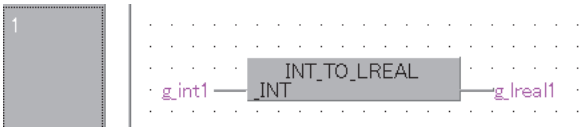
INT_TO_LREAL(_E), DINT_TO_LREAL(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，倍精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数

〔構造化ラダー/FBD〕

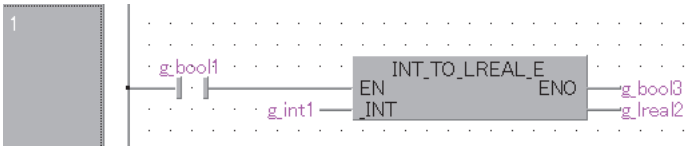


〔ST〕

```
g_lreal1 := INT_TO_LREAL(g_int1);
```

(b) EN/ENO 付き関数

〔構造化ラダー/FBD〕



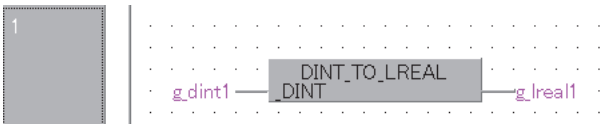
〔ST〕

```
g_bool3 := INT_TO_LREAL_E(g_bool1, g_int1, g_lreal2);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，倍精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数

〔構造化ラダー/FBD〕



〔ST〕

```
g_lreal1 := DINT_TO_LREAL(g_dint1);
```

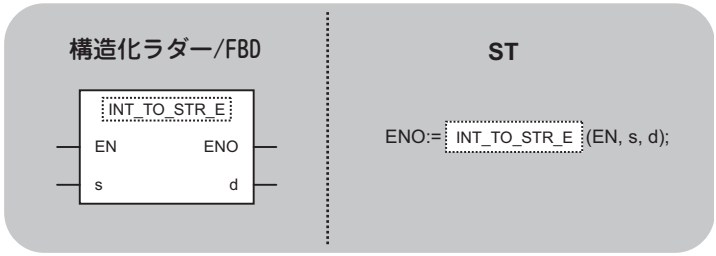
5.1.10 ワード［符号付き］型，ダブルワード［符号付き］型→文字列型変換

INT_TO_STR(_E), DINT_TO_STR(_E)

Basic / High performance / Process / Redundant / Universal / LCPU

INT_TO_STR(_E)
DINT_TO_STR(_E)

（ _E : EN/ENO 付き ）



[ ] には，下記の関数が入ります。

INT_TO_STR	INT_TO_STR_E
DINT_TO_STR	DINT_TO_STR_E

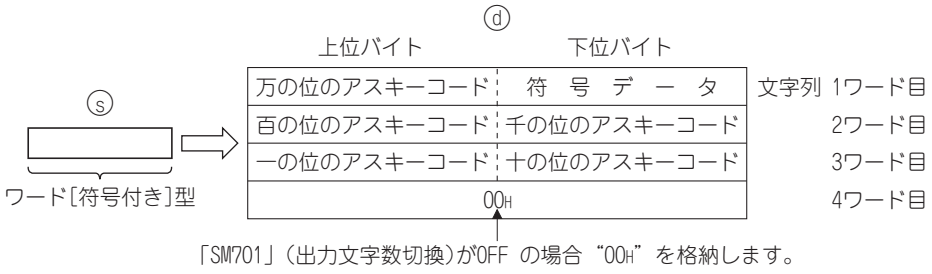
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_INT, _DINT):	入力	: ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 文字列 (6)/(11)

機　能

演算処理

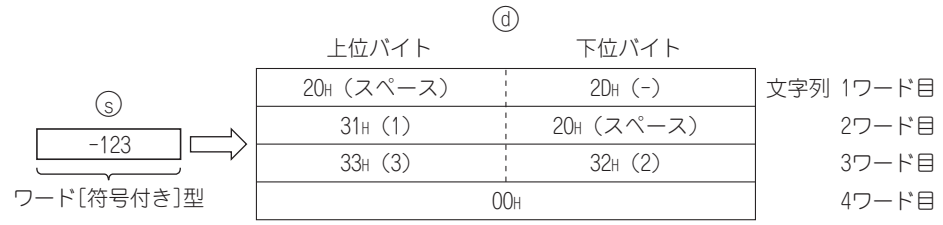
(1) INT_TO_STR, INT_TO_STR_E

(a) ⓢに入力されたワード［符号付き］型のデータを，文字列型のデータに変換してⓓから出力します。



(b) " 符号データ " には，入力された値が正のとき "20H（スペース）" を格納し，負のとき "2DH（–）" を格納します。

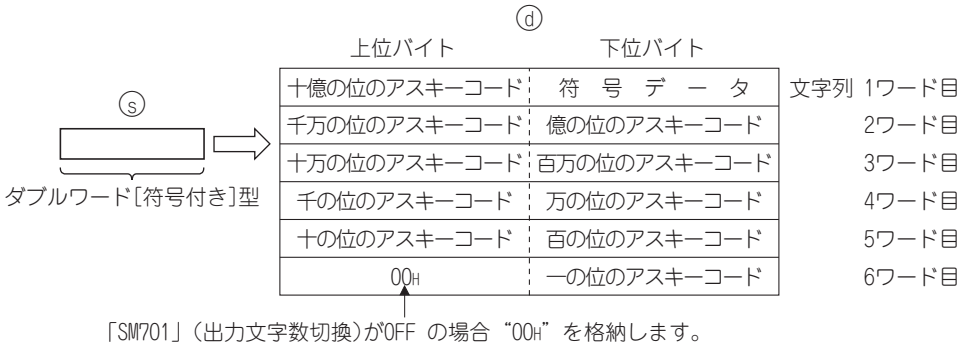
(c) 有効桁数が少ない場合は，上位桁に "20H（スペース）" を格納します。
（例）–123 を入力した場合



(d) SM701（出力文字数切換）が OFF の場合，文字列の最後に "00H" を格納します。

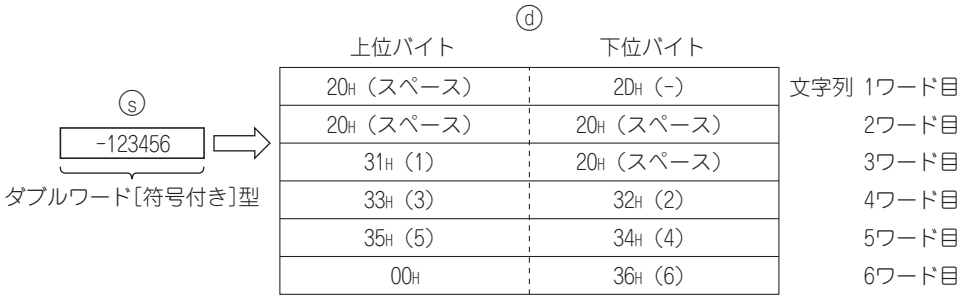
(2) DINT_TO_STR, DINT_TO_STR_E

(a) ⓢに入力されたダブルワード [符号付き] 型のデータを，文字列型のデータに変換してⓓから出力します。



(b) " 符号データ " には，入力された値が正のとき "20H（スペース）" を格納し，負のとき "2DH（–）" を格納します。

(c) 有効桁数が少ない場合は，上位桁に "20H（スペース）" を格納します。
（例）–123456 を入力した場合



(d) SM701（出力文字数切換）が OFF の場合，文字列の最後に "00H" を格納します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

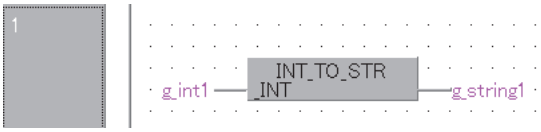
INT_TO_STR(_E), DINT_TO_STR(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_STR)

〔構造化ラダー/FBD〕

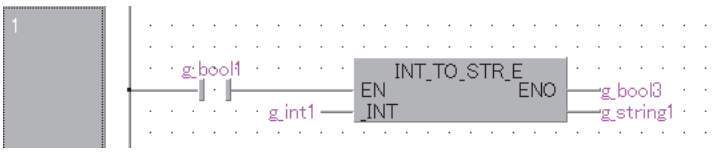


〔ST〕

```
g_string1 := INT_TO_STR(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_STR_E)

〔構造化ラダー/FBD〕



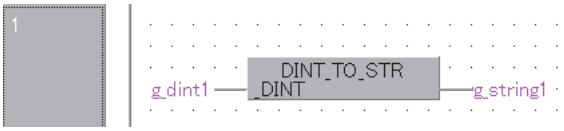
〔ST〕

```
g_bool3 := INT_TO_STR_E (g_bool1, g_int1, g_string1);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_STR)

〔構造化ラダー/FBD〕



〔ST〕

```
g_string1 := DINT_TO_STR (g_dint1);
```

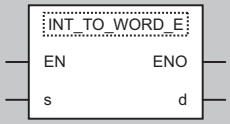
5.1.11 ワード［符号付き］型，ダブルワード［符号付き］型→ワード［符号なし］/ビット列［16 ビット］型変換 INT_TO_WORD(_E), DINT_TO_WORD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_WORD(_E)
DINT_TO_WORD(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD



ST

```
ENO:= INT_TO_WORD_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

INT_TO_WORD	INT_TO_WORD_E
DINT_TO_WORD	DINT_TO_WORD_E

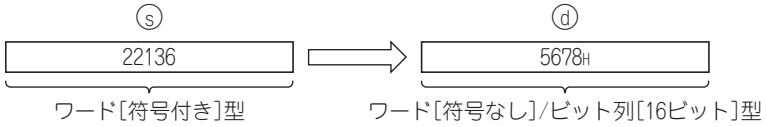
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_INT, _DINT):	入力	: ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号なし］/ ビット列［16 ビット］

機　能

演算処理

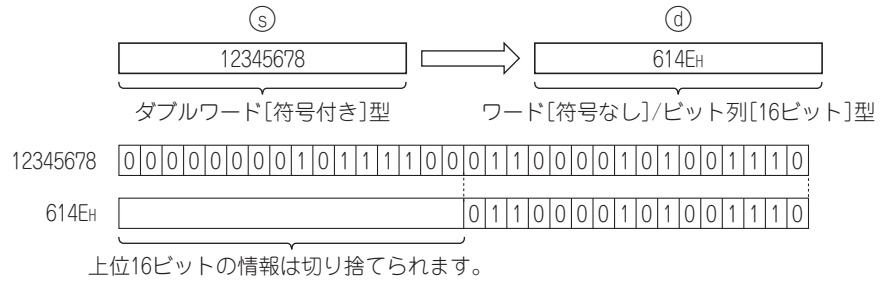
(1) INT_TO_WORD, INT_TO_WORD_E

ⓢに入力されたワード［符号付き］型のデータを，ワード［符号なし］/ ビット列［16 ビット］型のデータに変換してⓓから出力します。



(2) DINT_TO_WORD, DINT_TO_WORD_E

ⓢに入力されたダブルワード [符号付き] 型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

ポイント

DINT_TO_WORD(_E) を実行時，ⓢに入力されたダブルワード [符号付き] 型データ値の下位 16 ビットの値のみワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換されます。上位 16 ビットの情報は切り捨てられます。

エラー

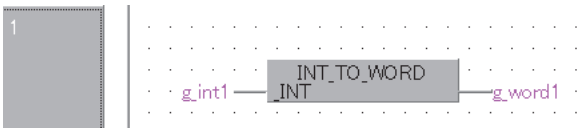
INT_TO_WORD(_E)，DINT_TO_WORD(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_WORD)

〔構造化ラダー/FBD〕

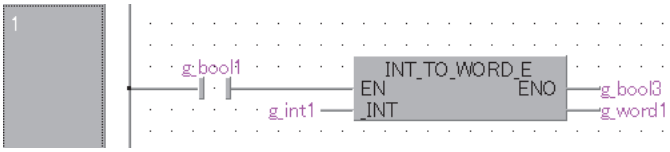


〔ST〕

```
g_word1 := INT_TO_WORD(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_WORD_E)

〔構造化ラダー/FBD〕



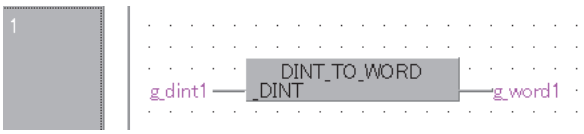
〔ST〕

```
g_bool3 := INT_TO_WORD_E (g_bool1, g_int1, g_word1);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_WORD)

〔構造化ラダー/FBD〕



〔ST〕

```
g_word1 := DINT_TO_WORD(g_dint1);
```

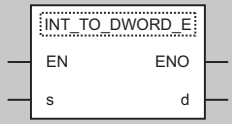
5.1.12 ワード［符号付き］型，ダブルワード［符号付き］型→ダブルワード［符号なし］/ ビット列［32 ビット］型変換 INT_TO_DWORD(_E), DINT_TO_DWORD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_DWORD(_E)
DINT_TO_DWORD(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= INT_TO_DWORD_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

INT_TO_DWORD	INT_TO_DWORD_E
DINT_TO_DWORD	DINT_TO_DWORD_E

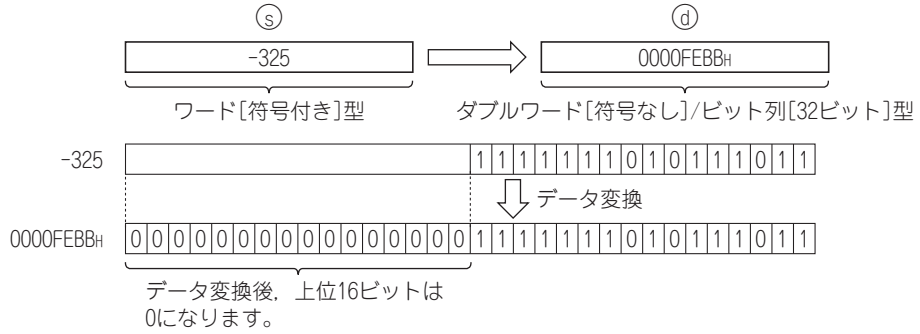
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_INT, _DINT):	入力	: ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ダブルワード［符号なし］/ ビット列［32 ビット］

機　能

演算処理

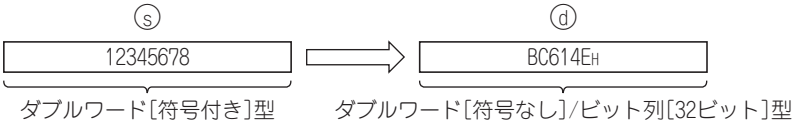
(1) INT_TO_DWORD, INT_TO_DWORD_E

ⓢに入力されたワード［符号付き］型のデータを，ダブルワード［符号なし］/ ビット列［32 ビット］型のデータに変換してⓓから出力します。



(2) DINT_TO_DWORD, DINT_TO_DWORD_E

ⓢに入力されたダブルワード [符号付き] 型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

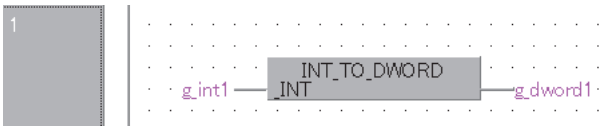
INT_TO_DWORD(_E), DINT_TO_DWORD(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_DWORD)

〔構造化ラダー/FBD〕

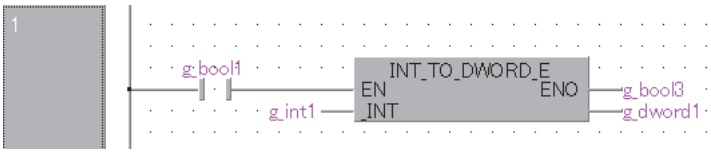


〔ST〕

```
g_dword1:= INT_TO_DWORD(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_DWORD_E)

〔構造化ラダー/FBD〕



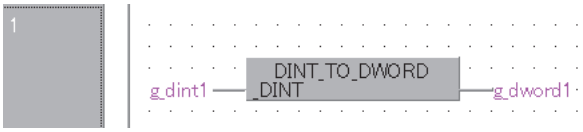
〔ST〕

```
g_bool3 := INT_TO_DWORD_E(g_bool1, g_int1, g_dword1);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_DWORD)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dword1:= DINT_TO_DWORD(g_dint1);
```

5.1.13 ワード［符号付き］型，ダブルワード［符号付き］型→BCD 型変換

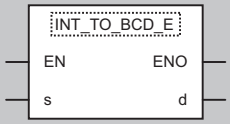
INT_TO_BCD(_E), DINT_TO_BCD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_BCD(_E)
DINT_TO_BCD(_E)

（ _E : EN/ENO 付き ）

構造化ラダー/FBD



ST

ENO:= INT_TO_BCD_E (EN, s, d);

［　　］には，下記の関数が入ります。

INT_TO_BCD	INT_TO_BCD_E
DINT_TO_BCD	DINT_TO_BCD_E

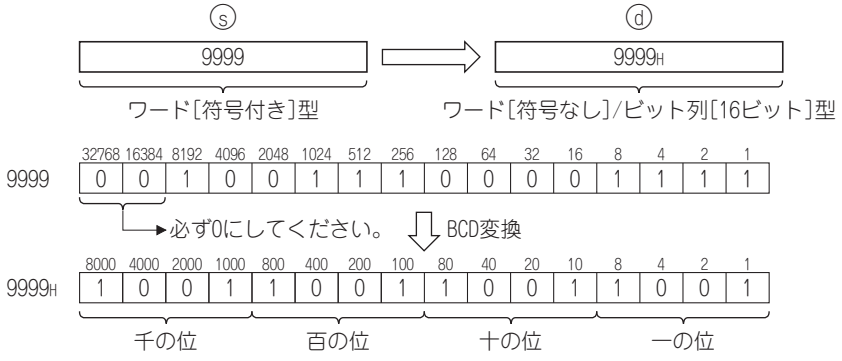
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_INT, _DINT):	入力	: ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］

機　能

演算処理

(1) INT_TO_BCD, INT_TO_BCD_E

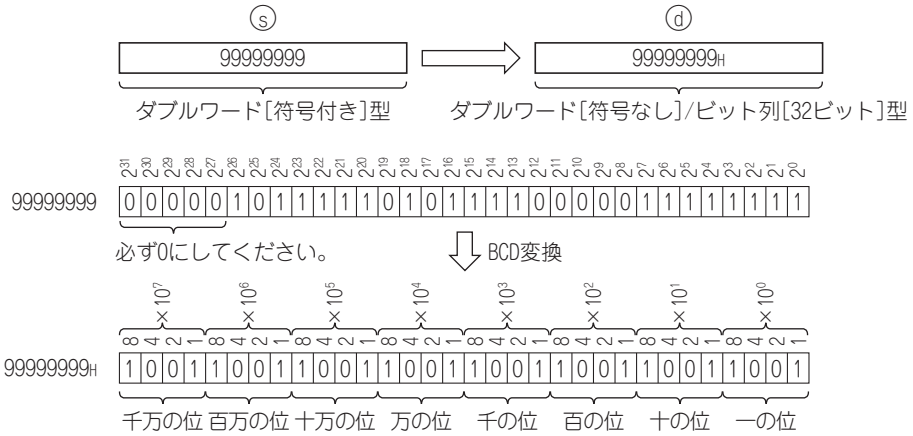
(a) ⓢに入力されたワード［符号付き］型のデータを，BCD 型のデータに変換してⓓから出力します。



(b) ⓢへの入力値は，ワード［符号付き］型のデータ値で 0 ～ 9999 の範囲内です。

(2) DINT_TO_BCD, DINT_TO_BCD_E

(a) ⓢに入力されたダブルワード [符号付き] 型のデータを，BCD 型のデータに変換してⓓから出力します。

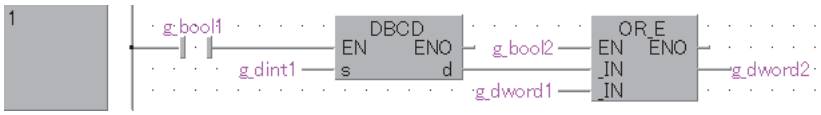


(b) ⓢへの入力値は，ダブルワード [符号付き] 型のデータ値で 0 ～ 99999999 の範囲内です。

(c) ⓓには，ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型が指定できます。ビット型は指定できません。

ポイント

ⓓからの出力は，ファンクションおよびオペレータのダブルワード [符号なし] / ビット列 [32 ビット] 型の入力に接続して使用できません。この場合は，DBCD 命令を使用してください。



演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

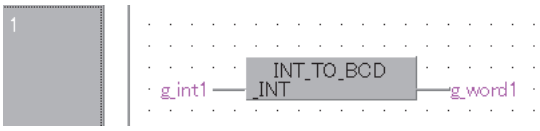
INT_TO_BCD(_E)，DINT_TO_BCD(_E) で入力された値がそれぞれ 9999，99999999 を超えた場合には演算エラーとなります。　（エラーコード：4100）

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，BCD 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_BCD)

〔構造化ラダー/FBD〕

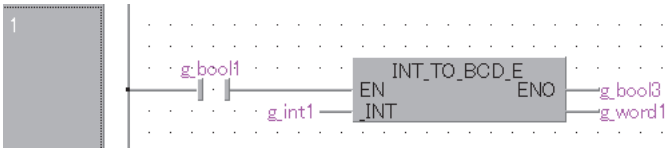


〔ST〕

```
g_word1:= INT_TO_BCD(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_BCD_E)

〔構造化ラダー/FBD〕



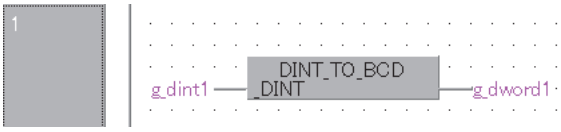
〔ST〕

```
g_bool3 := INT_TO_BCD_E(g_bool1, g_int1, g_word1);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，BCD 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_BCD)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dword1:= DINT_TO_BCD(g_dint1);
```

5.1.14 ワード［符号付き］型，ダブルワード［符号付き］型→時間型変換

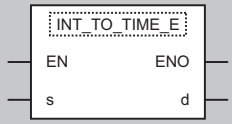
INT_TO_TIME(_E), DINT_TO_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_TIME(_E)
DINT_TO_TIME(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD



ST

```
ENO:= INT_TO_TIME_E (EN, s, d);
```

［　　］には，下記の関数が入ります。

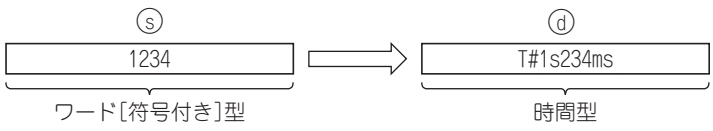
INT_TO_TIME	INT_TO_TIME_E
DINT_TO_TIME	DINT_TO_TIME_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s(_INT, _DINT):	入力	：ワード［符号付き］，ダブルワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	：ビット
	d:	出力	：時間

機　能

演算処理

ⓢに入力されたワード［符号付き］型／ダブルワード［符号付き］型のデータを，時間型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

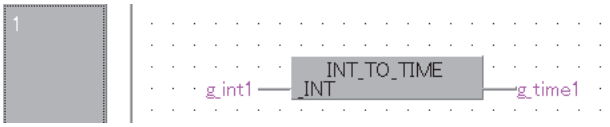
INT_TO_TIME(_E), DINT_TO_TIME(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型のデータを，時間型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_TIME)

〔構造化ラダー/FBD〕

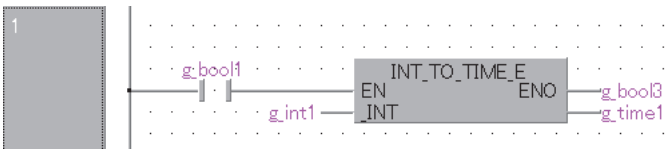


〔ST〕

```
g_time1:= INT_TO_TIME(g_int1);
```

(b) EN/ENO 付き関数 (INT_TO_TIME_E)

〔構造化ラダー/FBD〕



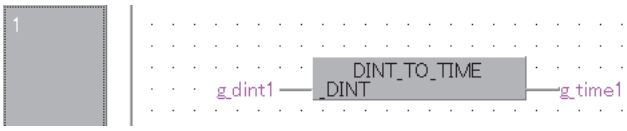
〔ST〕

```
g_bool3 := INT_TO_TIME_E(g_bool1, g_int1, g_time1);
```

(2) ⓢに入力されたダブルワード [符号付き] 型のデータを，時間型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DINT_TO_TIME)

〔構造化ラダー/FBD〕



〔ST〕

```
g_time1:= DINT_TO_TIME(g_dint1);
```

5.1.15 単精度実数型→ワード［符号付き］型，ダブルワード［符号付き］型変換

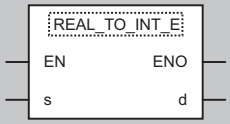
REAL_TO_INT(_E), REAL_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

REAL_TO_INT(_E)
REAL_TO_DINT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= REAL_TO_INT_E(EN, s, d);
```

[　　] には，下記の関数が入ります。

REAL_TO_INT	REAL_TO_INT_E
REAL_TO_DINT	REAL_TO_DINT_E

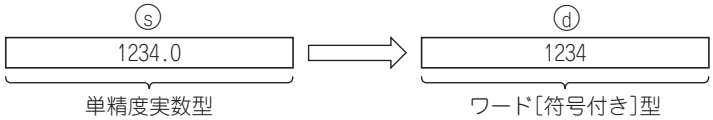
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_REAL):	入力	: 単精度実数
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号付き］，ダブルワード［符号付き］

機　能

演算処理

(1) REAL_TO_INT, REAL_TO_INT_E

(a) ⓢに入力された単精度実数型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。

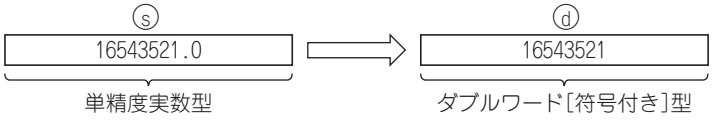


(b) ⓢへの入力値は，単精度実数型のデータ値で –32768 ～ 32767 の範囲内です。

(c) 変換後のデータは，単精度実数型データ値の小数点以下 1 桁目を四捨五入した値となります。

(2) REAL_TO_DINT, REAL_TO_DINT_E

(a) ⓢに入力された単精度実数型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。



(b) ⓢへの入力値は，単精度実数型のデータ値で –2147483648 ～ 2147483647 の範囲内です。
ただし，プログラミングツールから入力値を設定する場合，丸め誤差が発生する場合があります。プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

(c) 変換後のデータは，単精度実数型データ値の小数点以下 1 桁目を四捨五入した値となります。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

次の場合には演算エラーとなります。

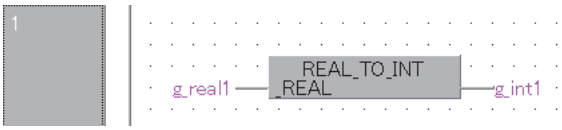
- REAL_TO_INT(_E)：入力された値が –32768 ～ 32767 以外の場合
  （エラーコード 4100）
- REAL_TO_DINT(_E)：入力された値が –2147483648 ～ 2147483647 以外の場合
  （エラーコード 4100）

プログラム例

(1) ⓢに入力された単精度実数型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (REAL_TO_INT)

〔構造化ラダー/FBD〕

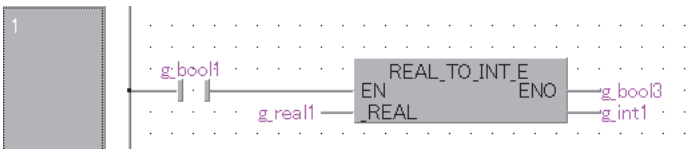


〔ST〕

```
g_int1:= REAL_TO_INT(g_real1);
```

(b) EN/ENO 付き関数 (REAL_TO_INT_E)

〔構造化ラダー/FBD〕



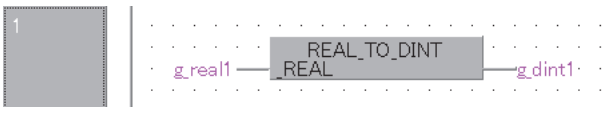
〔ST〕

```
g_bool3 := REAL_TO_INT_E(g_bool1, g_real1, g_int1);
```

(2) ⓢに入力された単精度実数型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (REAL_TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

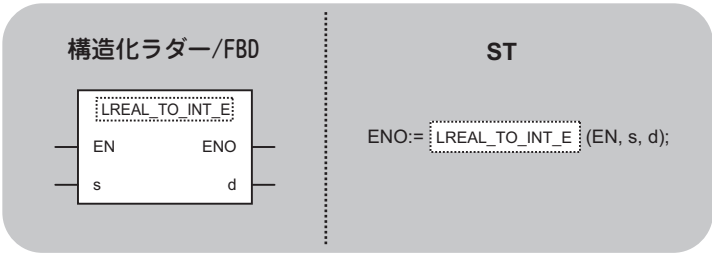
```
g_dint1:= REAL_TO_DINT(g_real1);
```

5.1.16 倍精度実数型→ワード［符号付き］型，ダブルワード［符号付き］型変換

LREAL_TO_INT(_E), LREAL_TO_DINT(_E)

LREAL_TO_INT(_E)
LREAL_TO_DINT(_E)

（ _E : EN/ENO 付き ）



[ ] には，下記の関数が入ります。

LREAL_TO_INT	LREAL_TO_INT_E
LREAL_TO_DINT	LREAL_TO_DINT_E

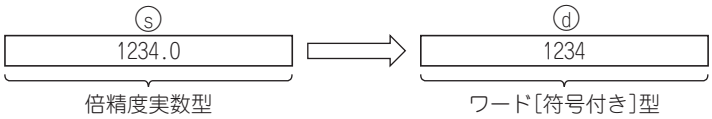
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_LREAL):	入力	: 倍精度実数
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号付き］，ダブルワード［符号付き］

機　能

演算処理

(1) LREAL_TO_INT, LREAL_TO_INT_E

(a) ⓢに入力された倍精度実数型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。

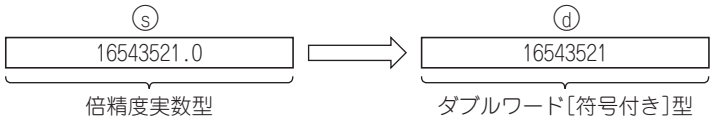


(b) ⓢへの入力値は，倍精度実数型のデータ値で –32768 ～ 32767 の範囲内です。

(c) 変換後のデータは，倍精度実数型データ値の小数点以下 1 桁目を四捨五入した値となります。

(2) LREAL_TO_DINT, LREAL_TO_DINT_E

(a) ⓢに入力された倍精度実数型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。



(b) ⓢへの入力値は，倍精度実数型のデータ値で –2147483648 ～ 2147483647 の範囲内です。ただし，プログラミングツールから入力値を設定する場合，丸め誤差が発生する場合があります。プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

(c) 変換後のデータは，倍精度実数型データ値の小数点以下 1 桁目を四捨五入した値となります。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

次の場合には演算エラーとなります。

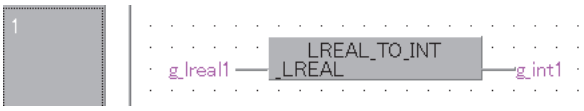
- 入力された値が -0，または下記範囲以外のとき。　（エラーコード 4140）
  0，$2^{-1022} \leqq |$ ⓢ $| < 2^{1024}$
- LREAL_TO_INT(_E)：入力された値が –32768.0 ～ 32767.0 以外の場合
  （エラーコード 4140）
- LREAL_TO_DINT(_E)：入力された値が –2147483648.0 ～ 2147483647.0 以外の場合
  （エラーコード 4140）

プログラム例

(1) ⓢに入力された倍精度実数型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数

〔構造化ラダー/FBD〕

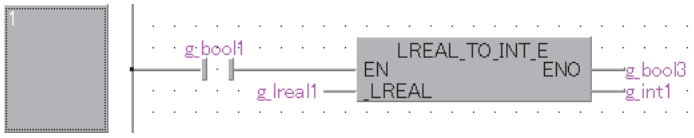


〔ST〕

```
g_int1:= LREAL_TO_INT(g_lreal1);
```

(b) EN/ENO 付き関数

〔構造化ラダー/FBD〕



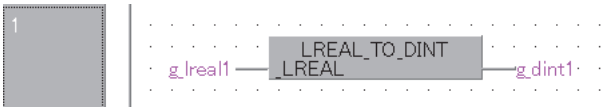
〔ST〕

```
g_bool3 := LREAL_TO_INT_E(g_bool1, g_lreal1, g_int1);
```

(2) ⓢに入力された倍精度実数型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint1:= LREAL_TO_DINT(g_lreal1);
```

5.1.17 単精度実数型→倍精度実数型変換

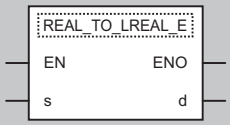
REAL_TO_LREAL(_E)

Basic × High performance × Process × Redundant × Universal LCPU

REAL_TO_LREAL(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= REAL_TO_LREAL_E (EN, s, d);

[ ] には，下記の関数が入ります。

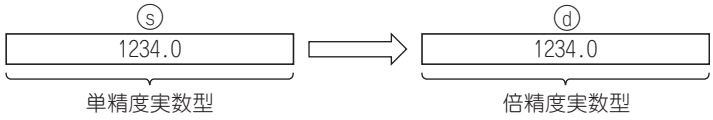
REAL_TO_LREAL	REAL_TO_LREAL_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_REAL):	入力	:単精度実数
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:倍精度実数

機　能

演算処理

(1) ⓢに入力された単精度実数型のデータを，倍精度実数型のデータに変換してⓓから出力します。



(2) プログラミングツールからⓢへの入力値を設定する場合，丸め誤差が発生する場合があります。プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

次の場合には演算エラーとなります。

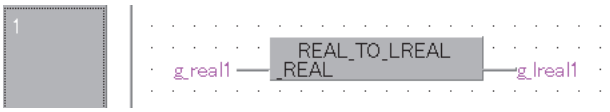
- 入力された値が -0，または下記範囲以外のとき。（エラーコード 4140）
  0，$2^{-126} \leqq |$ ⓢ $| < 2^{128}$
- 演算結果が下記範囲を超えるとき（オーバーフローが発生したとき）。
  $2^{1024} \leqq$ | 演算結果 |（エラーコード：4141）

プログラム例

ⓢに入力された単精度実数型のデータを，倍精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数

〔構造化ラダー/FBD〕

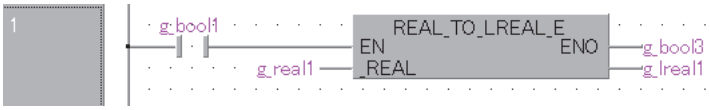


〔ST〕

```
g_lreal1:= REAL_TO_LREAL(g_real1);
```

(b) EN/ENO 付き関数

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := REAL_TO_LREAL_E(g_bool1, g_real1, g_lreal1);
```

5.1.18 倍精度実数型→単精度実数型変換

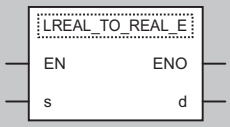
LREAL_TO_REAL(_E)

Basic (×) High performance (×) Process (×) Redundant (×) Universal LCPU

LREAL_TO_REAL(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= LREAL_TO_REAL_E (EN, s, d);

[ ] には，下記の関数が入ります。

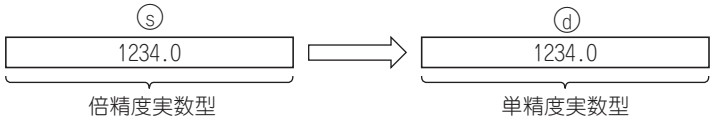
LREAL_TO_REAL	LREAL_TO_REAL_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_LREAL):	入力	:倍精度実数
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:単精度実数

機　能

演算処理

(1) ⓢに入力された倍精度実数型のデータを，単精度実数型のデータに変換してⓓから出力します。



(2) プログラミングツールからⓢへの入力値を設定する場合，丸め誤差が発生する場合があります。プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

次の場合には演算エラーとなります。

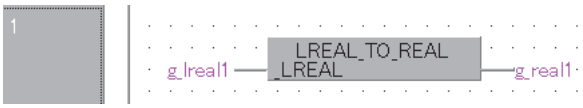
- 入力された値が -0，または下記範囲以外のとき。（エラーコード 4140）
  0，$2^{-1022} \leqq |$ ⓢ $| < 2^{1024}$
- 演算結果が下記範囲を超えるとき（オーバーフローが発生したとき）。
  $2^{128} \leqq$ | 演算結果 |（エラーコード 4141）

プログラム例

ⓢに入力された倍精度実数型のデータを，単精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数

〔構造化ラダー/FBD〕

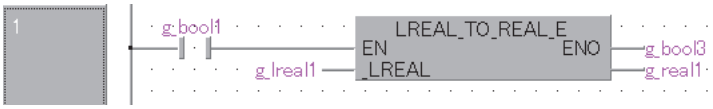


〔ST〕

```
g_real1:= LREAL_TO_REAL(g_lreal1);
```

(b) EN/ENO 付き関数

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := LREAL_TO_REAL_E(g_bool1, g_lreal1, g_real1);
```

5.1.19 単精度実数型→文字列型変換

REAL_TO_STR(_E)

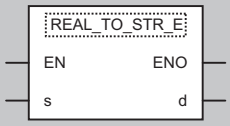
Ver. Basic / High performance / Process / Redundant / Universal / LCPU

シリアル No. の上 5 桁が “04122” 以降のベーシックモデル QCPU で使用可能。

REAL_TO_STR(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:=REAL_TO_STR_E(EN, s, d);
```

[ ] には，下記の関数が入ります。

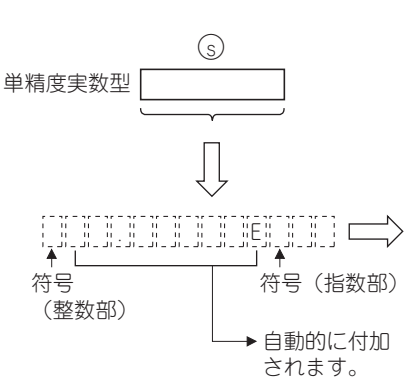
REAL_TO_STR　　REAL_TO_STR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_REAL):	入力	: 単精度実数
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 文字列 (13)

機　能

演算処理

(1) ⓢに入力された単精度実数型のデータを，文字列型（指数型式）のデータに変換してⓓから出力します。

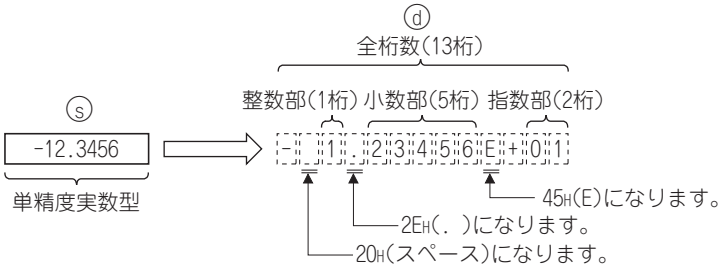


ⓓ 上位バイト	下位バイト	
20H（スペース）	符号データ（整数部）	文字列 1ワード目
2EH（.）	整数部のアスキーコード	2ワード目
小数点第二位のアスキーコード	小数点第一位のアスキーコード	3ワード目
小数点第四位のアスキーコード	小数点第三位のアスキーコード	4ワード目
45H（E）	小数点第五位のアスキーコード	5ワード目
指数部十の位のアスキーコード	符号データ（指数部）	6ワード目
00H（NUL）	指数部一の位のアスキーコード	7ワード目

「SM701」（出力文字数切換）がOFF の場合 “00H” を格納します。

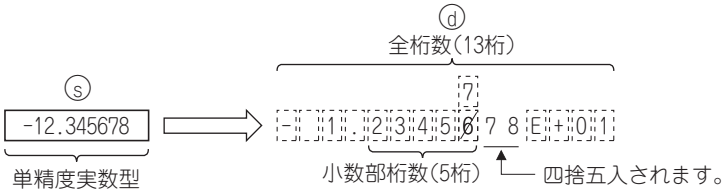
(2) 変換後の文字列データは，ⓓから次のように出力します。

(a) 整数部，小数部，指数部の桁数は，固定です。(整数部：1 桁，小数部：5 桁，指数部：2 桁)
2 バイト目，4 バイト目，10 バイト目には，それぞれ自動的に "20H"(スペース)，"2EH"(.) と "45H"(E) を格納します。

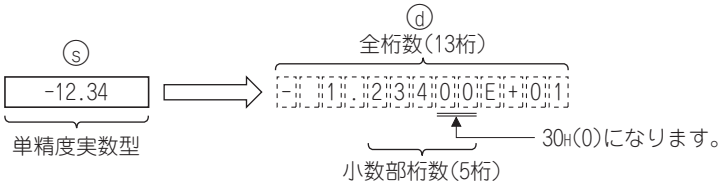


(b) 符号データ（整数部）には，入力された値が正のとき "20H"（スペース）を格納し，負のとき "2DH"（–）を格納します。

(c) 小数部の 6 桁目以降は，四捨五入されます。

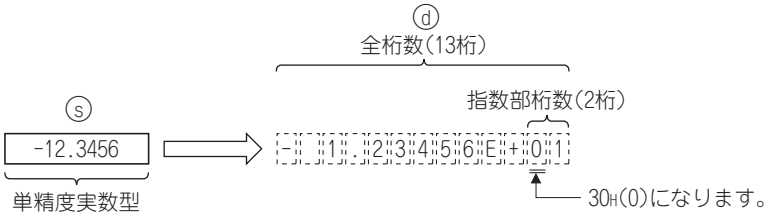


(d) 有効桁数が少ない場合は，小数部に "30H"(0) を格納します。



(e) 符号データ（指数部）には，指数が正のとき "2BH"（+）を格納し，負のとき "2DH"（–）を格納します。

(f) 指数部が 1 桁の場合は，指数部十の位へ "30H"(0) を格納します。



(3) 文字列の最後（7 ワード目）には，SM701(出力文字数切換) が OFF の場合，"00H" を格納します。

(4) プログラミングツールからⓢへの入力値を設定する場合，丸め誤差が発生する場合があります。プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

下記の場合は演算エラーとなります。

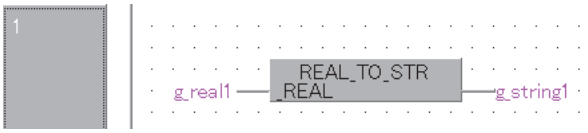
- 入力された値が -3.40282^{+38} ～ -1.17549^{-38}, 0, または 1.17549^{-38} ～ 3.40282^{+38} 以外のとき。（エラーコード 4100）

プログラム例

ⓢに入力された単精度実数型のデータを，文字列型（指数型式）のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (REAL_TO_STR)

〔構造化ラダー/FBD〕

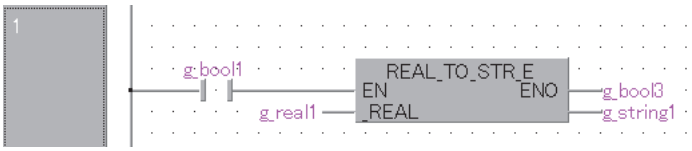


〔ST〕

```
g_string1:= REAL_TO_STR(g_real1);
```

(b) EN/ENO 付き関数 (REAL_TO_STR_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := REAL_TO_STR_E(g_bool1, g_real1, g_string1);
```

5.1.20 ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型→ビット型変換

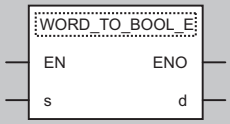
WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

WORD_TO_BOOL(_E)
DWORD_TO_BOOL(_E)

（ _E : EN/ENO 付き ）

構造化ラダー/FBD



ST

```
ENO:= WORD_TO_BOOL_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

WORD_TO_BOOL　WORD_TO_BOOL_E
DWORD_TO_BOOL　DWORD_TO_BOOL_E

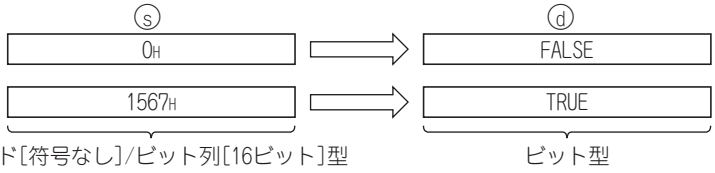
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s(_WORD,_DWORD):	入力	：ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	：ビット
	d:	出力	：ビット

機　能

演算処理

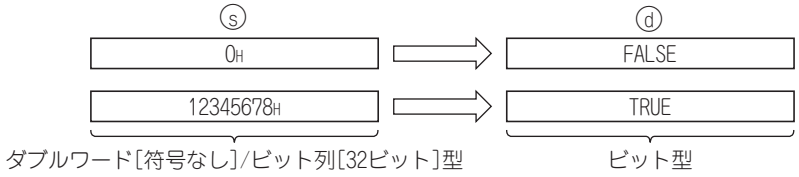
（1）WORD_TO_BOOL, WORD_TO_BOOL_E

ⓢに入力されたワード［符号なし］/ ビット列［16 ビット］型のデータを，ビット型のデータに変換してⓓから出力します。
入力値が 0H の場合は，FALSE を出力します。
入力値が 0H 以外の場合は，TRUE を出力します。



(2) DWORD_TO_BOOL, DWORD_TO_BOOL_E

ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ビット型のデータに変換してⓓから出力します。
入力値が 0H の場合は，FALSE を出力します。
入力値が 0H 以外の場合は，TRUE を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

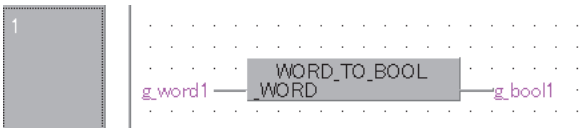
WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，ビット型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (WORD_TO_BOOL)

〔構造化ラダー/FBD〕

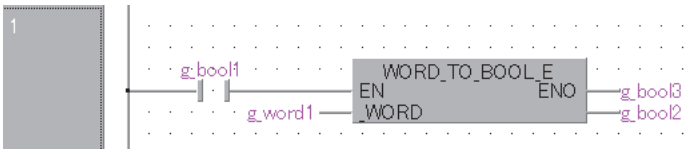


〔ST〕

```
g_bool1:= WORD_TO_BOOL(g_word1);
```

(b) EN/ENO 付き関数 (WORD_TO_BOOL_E)

〔構造化ラダー/FBD〕



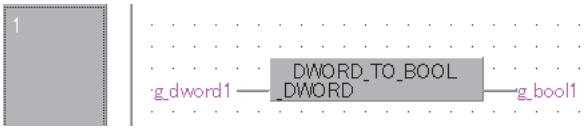
〔ST〕

```
g_bool3 := WORD_TO_BOOL_E(g_bool1, g_word1, g_bool2);
```

(2) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ビット型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DWORD_TO_BOOL)

〔構造化ラダー/FBD〕



〔ST〕

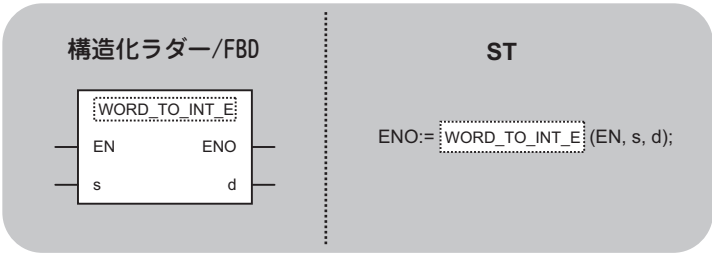
```
g_bool1:= DWORD_TO_BOOL(g_dword1);
```

5.1.21 ワード［符号なし］/ ビット列［16 ビット］型→ワード［符号付き］型，ダブルワード［符号付き］型変換 WORD_TO_INT(_E), WORD_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

WORD_TO_INT(_E)
WORD_TO_DINT(_E)

（ _E : EN/ENO 付き ）



[ ] には，下記の関数が入ります。

WORD_TO_INT	WORD_TO_INT_E
WORD_TO_DINT	WORD_TO_DINT_E

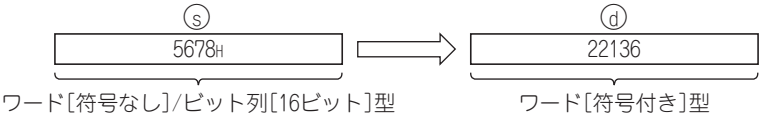
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_WORD):	入力	: ワード［符号なし］/ ビット列［16 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号付き］，ダブルワード［符号付き］

機　能

演算処理

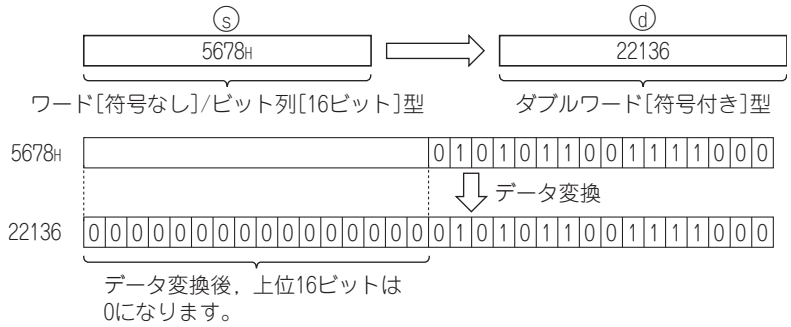
(1) WORD_TO_INT, WORD_TO_INT_E

ⓢに入力されたワード［符号なし］/ ビット列［16 ビット］型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。



(2) WORD_TO_DINT, WORD_TO_DINT_E

ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

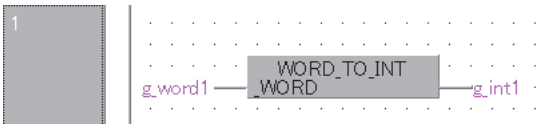
WORD_TO_INT(_E), WORD_TO_DINT(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (WORD_TO_INT)

〔構造化ラダー/FBD〕

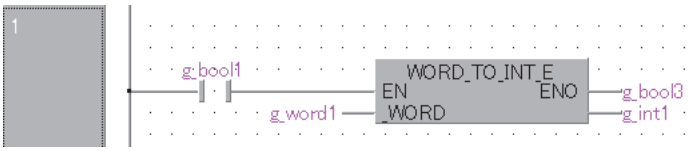


〔ST〕

```
g_int1:= WORD_TO_INT(g_word1);
```

(b) EN/ENO 付き関数 (WORD_TO_INT_E)

〔構造化ラダー/FBD〕



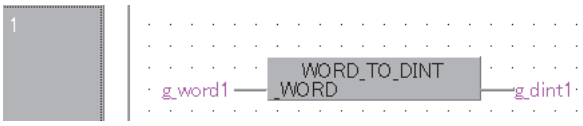
〔ST〕

```
g_bool3 := WORD_TO_INT_E(g_bool1, g_word1, g_int1);
```

(2) ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (WORD_TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

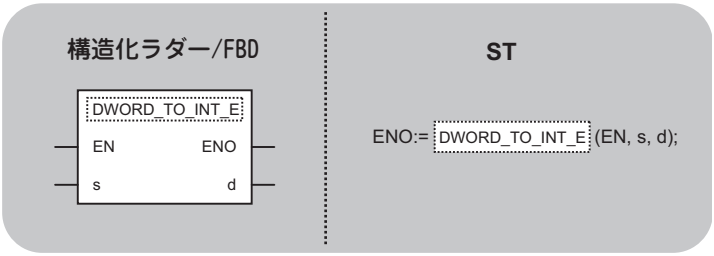
```
g_dint1:= WORD_TO_DINT(g_word1);
```

5.1.22 ダブルワード［符号なし］/ ビット列［32 ビット］型→ワード［符号付き］型，ダブルワード［符号付き］型変換 DWORD_TO_INT(_E), DWORD_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

DWORD_TO_INT(_E)
DWORD_TO_DINT(_E)

（ _E : EN/ENO 付き ）



[ ] には，下記の関数が入ります。

DWORD_TO_INT	DWORD_TO_INT_E
DWORD_TO_DINT	DWORD_TO_DINT_E

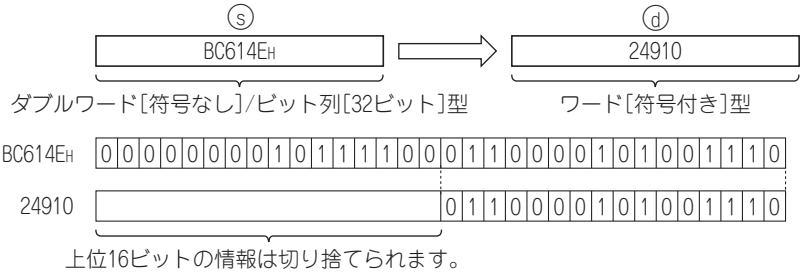
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_DWORD):	入力	: ダブルワード［符号なし］/ ビット列［32 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号付き］，ダブルワード［符号付き］

機　能

演算処理

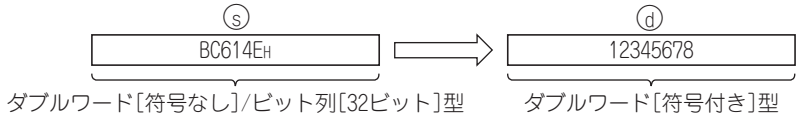
(1) DWORD_TO_INT, DWORD_TO_INT_E

ⓢに入力されたダブルワード［符号なし］/ ビット列［32 ビット］型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。



(2) DWORD_TO_DINT, DWORD_TO_DINT_E

ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

ポイント

DWORD_TO_INT(_E) を実行時，ⓢから入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型データ値の下位 16 ビットの値のみワード [符号付き] 型のデータに変換されます。上位 16 ビットの情報は切り捨てられます。

エラー

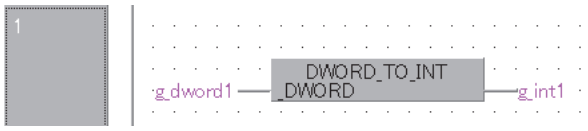
DWORD_TO_INT(_E), DWORD_TO_DINT(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DWORD_TO_INT)

〔構造化ラダー/FBD〕

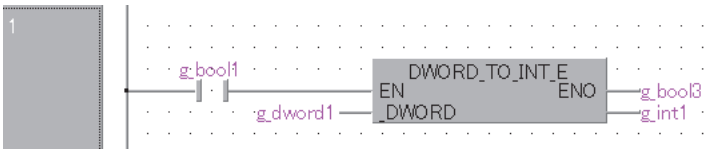


〔ST〕

```
g_int1:= DWORD_TO_INT(g_dword1);
```

(b) EN/ENO 付き関数 (DWORD_TO_INT_E)

〔構造化ラダー/FBD〕



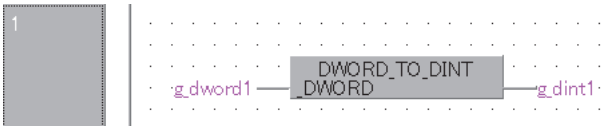
〔ST〕

```
g_bool3 := DWORD_TO_INT_E(g_bool1, g_dword1, g_int1);
```

(2) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DWORD_TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint1:= DWORD_TO_DINT(g_dword1);
```

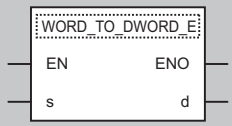
5.1.23 ワード［符号なし］/ ビット列［16 ビット］型→ダブルワード［符号なし］/ ビット列［32 ビット］型変換 WORD_TO_DWORD(_E)

Basic High performance Process Redundant Universal LCPU

WORD_TO_DWORD(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD



ST

```
ENO:= WORD_TO_DWORD_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

WORD_TO_DWORD　WORD_TO_DWORD_E

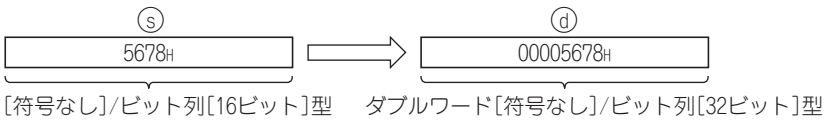
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_WORD):	入力	:ワード［符号なし］/ ビット列［16 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ダブルワード［符号なし］/ ビット列［32 ビット］

機　能

演算処理

ⓢに入力されたワード［符号なし］/ ビット列［16 ビット］型のデータを，ダブルワード［符号なし］/ ビット列［32 ビット］型のデータに変換してⓓから出力します。

データ変換後，上位 16 ビットは 0 となります。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

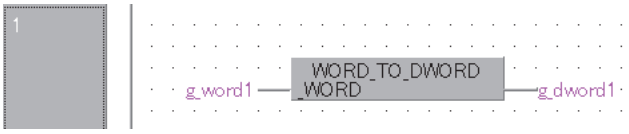
WORD_TO_DWORD(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (WORD_TO_DWORD)

〔構造化ラダー/FBD〕

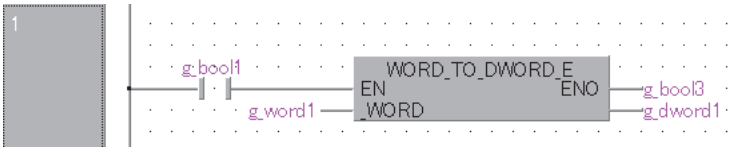


〔ST〕

```
g_dword1:= WORD_TO_DWORD(g_word1);
```

(b) EN/ENO 付き関数 (WORD_TO_DWORD_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := WORD_TO_DWORD_E(g_bool1, g_word1, g_dword1);
```

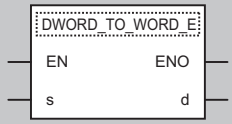
5.1.24 ダブルワード［符号なし］/ ビット列［32 ビット］型→ワード［符号なし］/ ビット列［16 ビット］型変換 DWORD_TO_WORD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

DWORD_TO_WORD(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= DWORD_TO_WORD_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

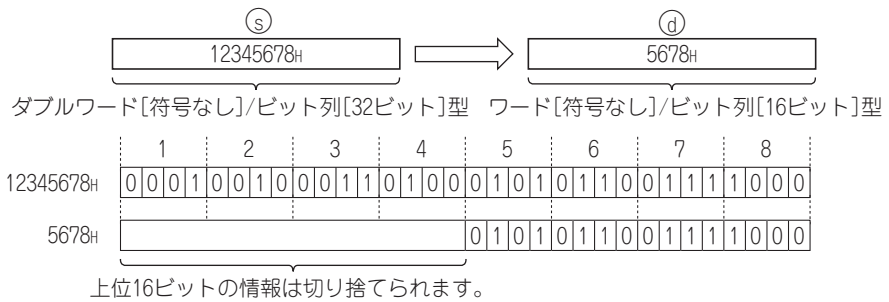
DWORD_TO_WORD　DWORD_TO_WORD_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_DWORD):	入力	:ダブルワード［符号なし］/ ビット列［32 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ワード［符号なし］/ ビット列［16 ビット］

機　能

演算処理

ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

ポイント

DWORD_TO_WORD(_E) を実行時，ⓢから入力されたダブルワード [符号なし]/ビット列 [32 ビット] 型データ値の下位 16 ビットの値のみワード [符号なし]/ビット列 [16 ビット] 型のデータに変換されます。
上位 16 ビットの情報は切り捨てられます。

エラー

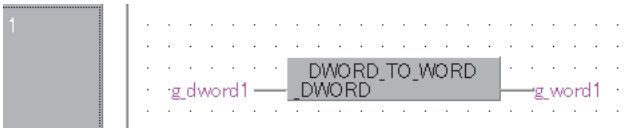
DWORD_TO_WORD(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DWORD_TO_WORD)

〔構造化ラダー/FBD〕

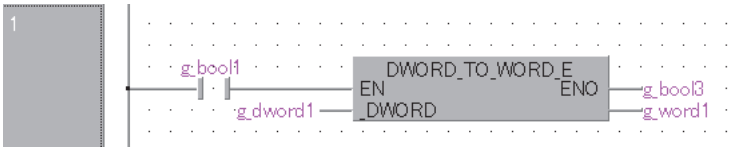


〔ST〕

```
g_word1:= DWORD_TO_WORD(g_dword1);
```

(b) EN/ENO 付き関数 (DWORD_TO_WORD_E)

〔構造化ラダー/FBD〕



〔ST〕

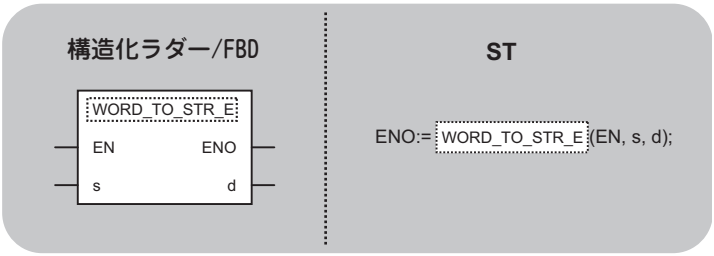
```
g_bool3 := DWORD_TO_WORD_E(g_bool1, g_dword1, g_word1);
```

5.1.25 ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型→文字列型変換

WORD_TO_STR(_E), DWORD_TO_STR(_E)

WORD_TO_STR(_E)
DWORD_TO_STR(_E)

（ _E : EN/ENO 付き ）



[ ] には，下記の関数が入ります。	
WORD_TO_STR	WORD_TO_STR_E
DWORD_TO_STR	DWORD_TO_STR_E

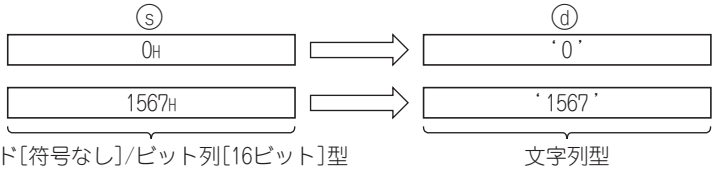
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_WORD, _DWORD):	入力	: ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 文字列 (4)/(8)

機　能

演算処理

(1) WORD_TO_STR, WORD_TO_STR_E

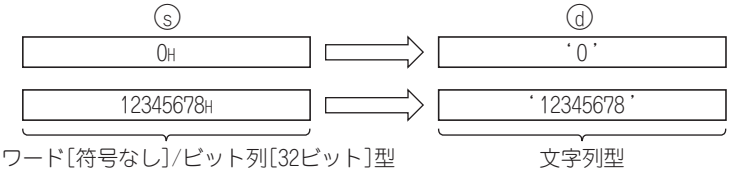
(a) ⓢに入力されたワード［符号なし］/ ビット列［16 ビット］型のデータを，文字列型のデータに変換してⓓから出力します。



(b) SM701（出力文字数切換）が OFF の場合，文字列の最後に "00H" を格納します。

(2) DWORD_TO_STR, DWORD_TO_STR_E

(a) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，文字列型のデータに変換してⓓから出力します。



(b) SM701（出力文字数切換）が OFF の場合，文字列の最後に "00H" を格納します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

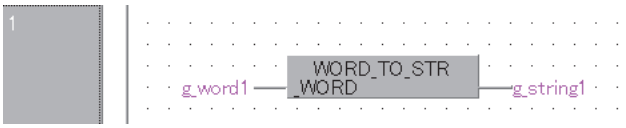
WORD_TO_STR(_E), DWORD_TO_STR(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (WORD_TO_STR)

〔構造化ラダー/FBD〕

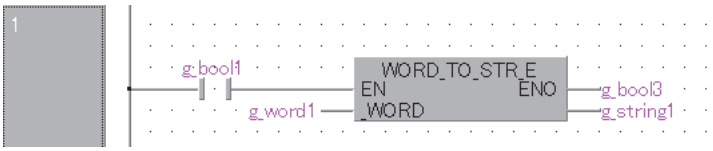


〔ST〕

```
g_string1 := WORD_TO_STR (g_word1);
```

(b) EN/ENO 付き関数 (WORD_TO_STR_E)

〔構造化ラダー/FBD〕



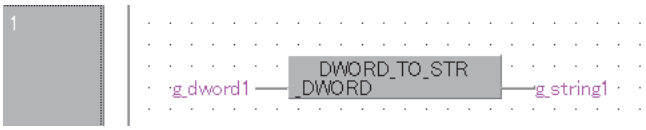
〔ST〕

```
g_bool3 := WORD_TO_STR_E (g_bool1, g_word1, g_string1);
```

(2) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DWORD_TO_STR)

〔構造化ラダー/FBD〕



〔ST〕

```
g_string1:= DWORD_TO_STR (g_dword1);
```

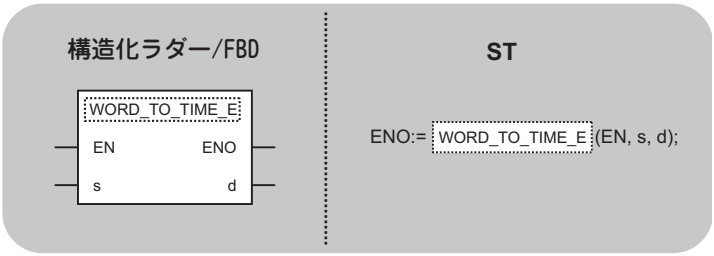
5.1.26 ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型→時間型変換

WORD_TO_TIME(_E), DWORD_TO_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

WORD_TO_TIME(_E)
DWORD_TO_TIME(_E)

（_E : EN/ENO 付き）



[ ] には，下記の関数が入ります。

WORD_TO_TIME	WORD_TO_ TIME _E
DWORD_TO_TIME	DWORD_TO_ TIME _E

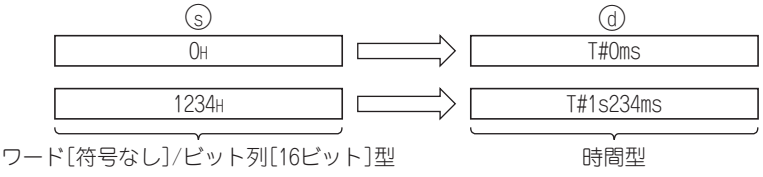
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_WORD, _DWORD):	入力	: ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 時間

機　能

演算処理

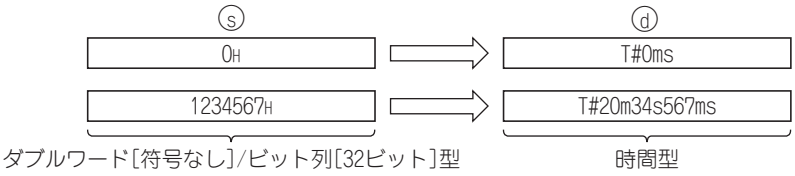
(1) WORD_TO_TIME, WORD_TO_TIME_E

ⓢに入力されたワード［符号なし］/ ビット列［16 ビット］型のデータを，時間型のデータに変換してⓓから出力します。



(2) DWORD_TO_TIME, DWORD_TO_TIME_E

ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，時間型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

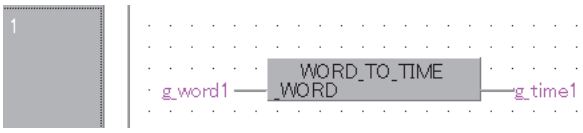
WORD_TO_TIME(_E), DWORD_TO_TIME(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，時間型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (WORD_TO_TIME)

〔構造化ラダー/FBD〕

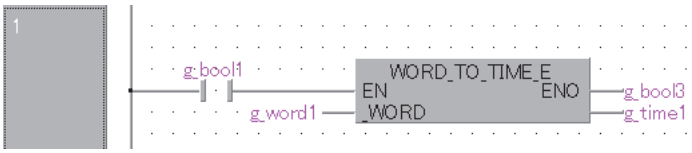


〔ST〕

```
g_time1 := WORD_TO_TIME (g_word1);
```

(b) EN/ENO 付き関数 (WORD_TO_TIME_E)

〔構造化ラダー/FBD〕



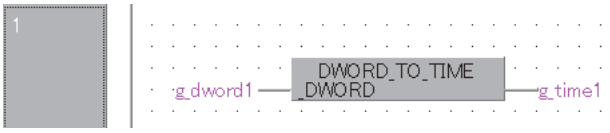
〔ST〕

```
g_bool3 := WORD_TO_TIME_E (g_bool1, g_word1, g_time1);
```

(2) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，時間型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (DWORD_TO_TIME)

〔構造化ラダー/FBD〕



〔ST〕

```
g_time1 := DWORD_TO_TIME (g_dword1)
```

5.1.27 文字列型→ビット型変換

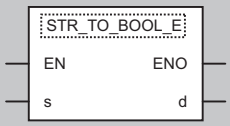
STR_TO_BOOL(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

STR_TO_BOOL(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= STR_TO_BOOL_E (EN, s, d);

[ ] には，下記の関数が入ります。

STR_TO_BOOL　　STR_TO_BOOL_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_STRING):	入力	:文字列(1)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ビット

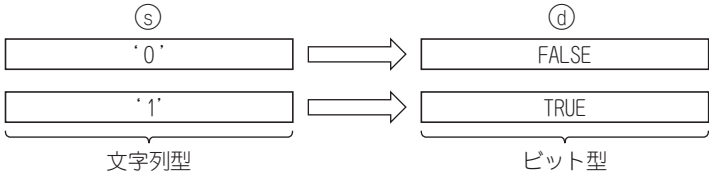
機　能

演算処理

ⓢに入力された文字列型のデータを，ビット型のデータに変換してⓓから出力します。

入力値が ‘0’ の場合は，FALSE を出力します。

入力値が ‘0’ 以外の場合は，TRUE を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

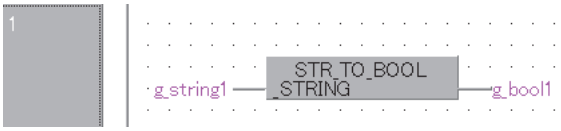
STR_TO_BOOL(_E) での演算エラーはありません。

プログラム例

ⓢに入力された文字列型のデータを，ビット型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_BOOL)

〔構造化ラダー/FBD〕

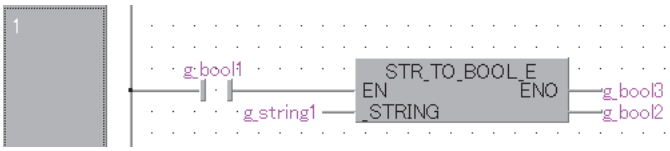


〔ST〕

```
g_bool1  := STR_TO_BOOL (g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_BOOL_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := STR_TO_BOOL_E (g_bool1, g_string1, g_bool2);
```

5.1.28 文字列型→ワード［符号付き］型，ダブルワード［符号付き］型変換

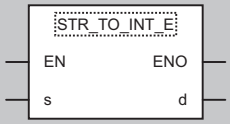
STR_TO_INT(_E), STR_TO_DINT(_E)

Basic High performance Process Redundant Universal LCPU

STR_TO_INT(_E)
STR_TO_DINT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= STR_TO_INT_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

STR_TO_INT	STR_TO_INT_E
STR_TO_DINT	STR_TO_DINT_E

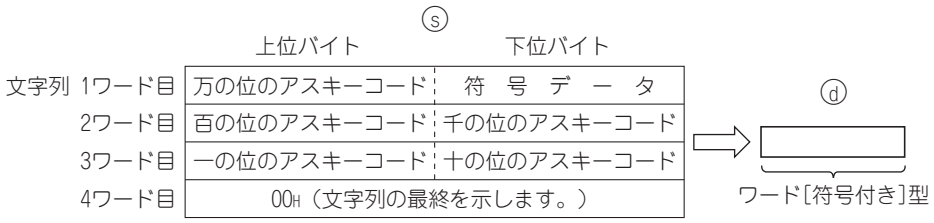
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_STRING):	入力	: 文字列 (6)/(11)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード [符号付き]，ダブルワード [符号付き]

機　能

演算処理

(1) STR_TO_INT, STR_TO_INT_E

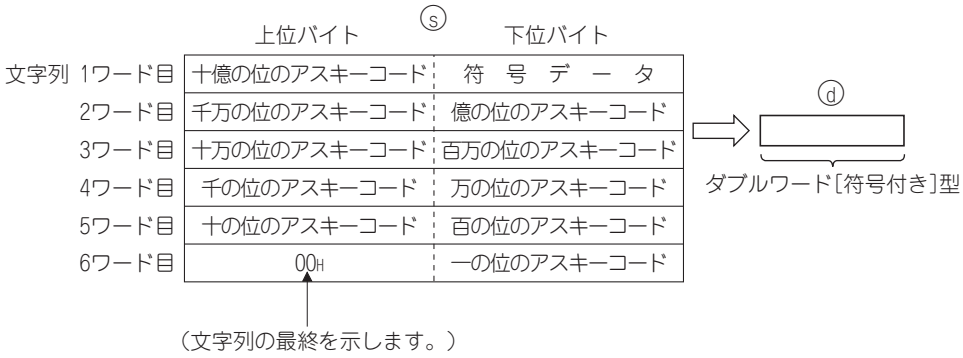
(a) ⓢに入力された文字列型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力します。



(b) ⓢへの入力値は，文字列型のデータ値で下記に示す範囲内です。
アスキーコードで "30H" ～ "39H"，"20H"，"2DH"，"00H" の範囲。
文字列型のデータ値で "–32768" ～ "32767" の範囲。

(2) STR_TO_DINT, STR_TO_DINT_E

(a) ⓢに入力された文字列型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。



(b) ⓢへの入力値は，文字列型のデータ値で下記に示す範囲内です。
アスキーコードで "30H" ～ "39H"，"20H"，"2DH"，"00H" の範囲。
文字列型のデータ値で –2147483648 ～ 2147483647 の範囲。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

下記の場合には演算エラーとなります。

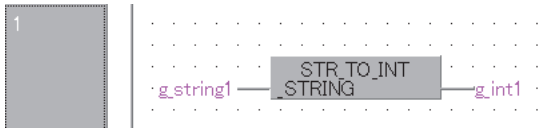
- 入力された値がアスキーコードで "30H" ～ "39H", "20H", "2DH", "00H" 以外のとき。 (エラーコード 4100)
- 入力された値がアスキーコードで下記の示す範囲外のとき。 (エラーコード 4100)
  STR_TO_INT(_E) : –32768 ～ 32767
  STR_TO_DINT(_E) : –2147483648 ～ 2147483647

プログラム例

(1) ⓢに入力された文字列型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_INT)

〔構造化ラダー/FBD〕

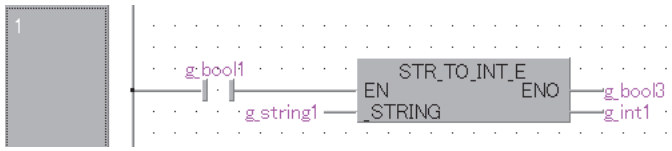


〔ST〕

```
g_int1 := STR_TO_INT (g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_INT_E)

〔構造化ラダー/FBD〕



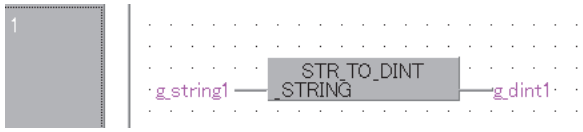
〔ST〕

```
g_bool3 := STR_TO_INT_E (g_bool1, g_string1, g_int1);
```

(2) ⓢに入力された文字列型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint1 := STR_TO_DINT (g_string1);
```

5.1.29 文字列型→単精度実数型変換

STR_TO_REAL(_E)

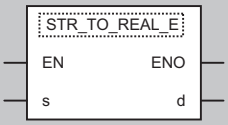
Ver. Basic / High performance / Process / Redundant / Universal / LCPU

シリアル No. の上 5 桁が “04122” 以降のベーシックモデル QCPU で使用可能。

STR_TO_REAL(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= STR_TO_REAL_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

STR_TO_REAL	STR_TO_REAL_E

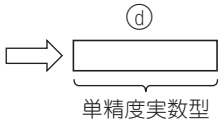
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_STRING):	入力	:文字列(24)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:単精度実数

機　能

演算処理

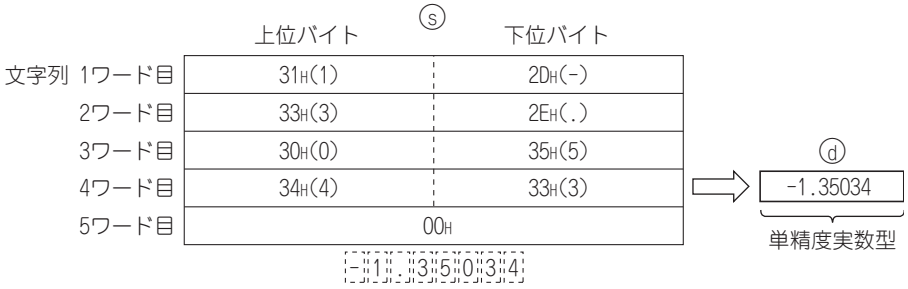
(1) ⓢに入力された文字列型（小数点型式／指数型式）のデータを，単精度実数型のデータに変換してⓓから出力します。

	ⓢ 上位バイト	下位バイト
文字列 1ワード目	1文字目のアスキーコード	符　号　デ　ー　タ
2ワード目	3文字目のアスキーコード	2文字目のアスキーコード
3ワード目	5文字目のアスキーコード	4文字目のアスキーコード
4ワード目	7文字目のアスキーコード	6文字目のアスキーコード
5ワード目	9文字目のアスキーコード	8文字目のアスキーコード
6ワード目	11文字目のアスキーコード	10文字目のアスキーコード
7ワード目	00H（文字列の最終を示します。）	

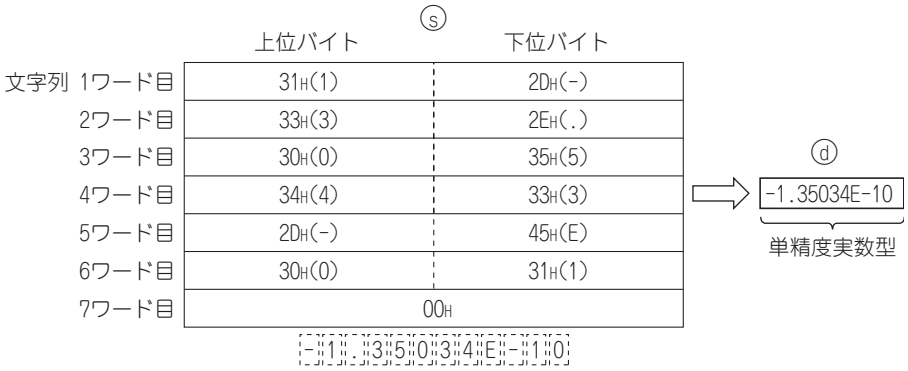


(2) 文字列型データは，小数点型式，指数型式どちらも変換できます。

(a) 小数点型式の場合



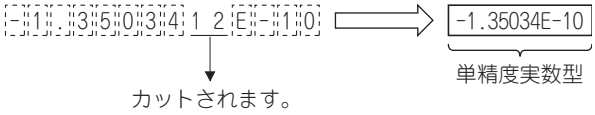
(b) 指数型式の場合



(3) 文字列型データは，符号，小数点，指数部を除いて 6 桁が有効となり，7 桁目以降はカットされて変換されます。

(a) 小数点型式の場合

(b) 指数型式の場合



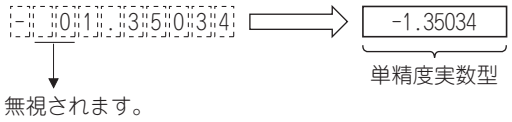
(4) 小数点型式で符号に "2BH"（+）を指定するか，符号を省略すると正の値として変換します。また，符号に "2DH"（–）を指定すると負の値として変換します。

(5) 指数型式で指数部の符号に "2BH"（+）を指定するか，符号を省略すると正の値として変換します。

指数部の符号に "2DH"（–）を指定すると負の値として変換します。

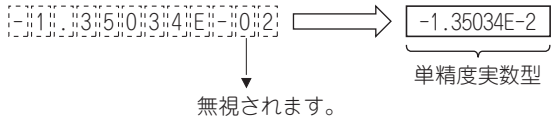
(6) 文字列型データで，最初の "0" 以外の数値の前に "20H"（スペース）または "30H"(0) が存在する場合は，"20H"，"30H" を無視して変換します。

(a) 小数点型式の場合



(b) 指数型式の場合

(7) 文字列型データ（指数型式）で "E" と数値の間に "30H(0)" が存在する場合は "30H" を無視して変換します。



(8) 文字列中に "20H"（スペース）が含まれていた場合は，"20H" を無視して変換します。

(9) 文字列型データは，最大 24 文字まで入力できます。
文字列中の "20H"（スペース），"30H"(0) も 1 文字としてカウントします。

(10) ⓢへの入力値は，文字列型のデータ値で下記に示す範囲内です。
アスキーコードで "30H" ～ "39H"，"45H"，"2BH"，"2DH"，"2EH"，"20H"，"00H" の範囲。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

下記の場合には演算エラーとなります。

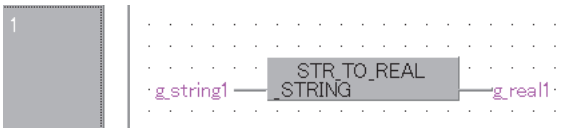
- 整数部，小数部に "30H" ～ "39H" 以外の文字があるとき。　（エラーコード 4100）
- 2EH が 2 つ以上あるとき。　（エラーコード 4100）
- 指数部に "45H"(E)，"2BH"(+)，"2DH"(–) 以外の文字があるとき，または指数部が複数あるとき。　（エラーコード 4100）
- 変換後のデータが -3.40282^{+38} ～ -1.17549^{-38}，0，または 1.17549^{-38} ～ 3.40282^{+38} 以外のとき。　（エラーコード 4100）
- 文字数が 0，または 24 文字を超えているとき。　（エラーコード 4100）

プログラム例

ⓢに入力された文字列型のデータを，単精度実数型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_REAL)

〔構造化ラダー/FBD〕

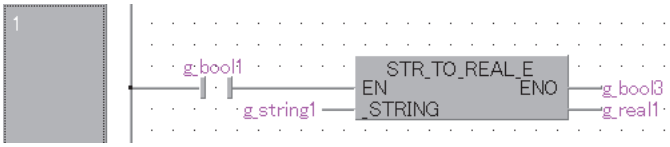


〔ST〕

```
g_real1 := STR_TO_REAL (g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_REAL_E)

〔構造化ラダー/FBD〕



〔ST〕

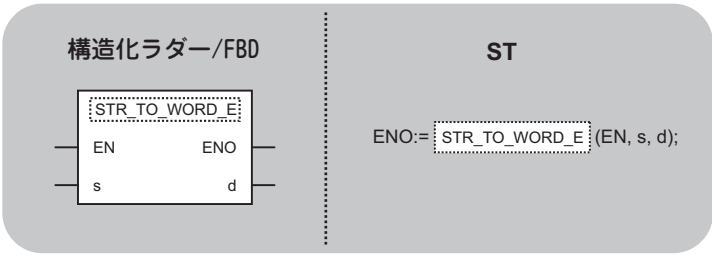
```
g_bool3  := STR_TO_REAL_E (g_bool1, g_string1, g_real1);
```

5.1.30 文字列型→ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型変換

STR_TO_WORD(_E), STR_TO_DWORD(_E)

STR_TO_WORD(_E)
STR_TO_DWORD(_E)

（ _E : EN/ENO 付き ）



には，下記の関数が入ります。	
STR_TO_WORD	STR_TO_WORD_E
STR_TO_DWORD	STR_TO_DWORD_E

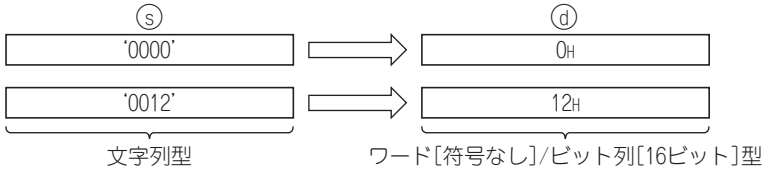
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_STRING):	入力	: 文字列 (4)/(8)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード [符号なし]/ ビット列 [16 ビット]，ダブルワード [符号なし]/ ビット列 [32 ビット]

機　能

演算処理

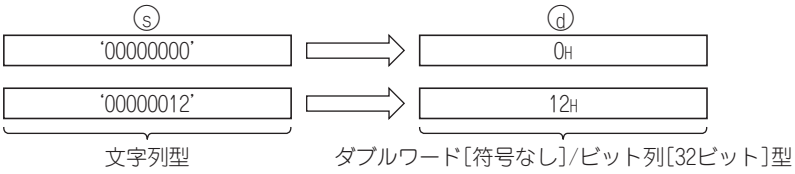
(1) STR_TO_WORD, STR_TO_WORD_E

ⓢに入力された文字列型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力します。



(2) STR_TO_DWORD, STR_TO_DWORD_E

ⓢに入力された文字列型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
STR_TO_WORD(_E) : HABIN
STR_TO_DWORD(_E) : DHABIN

次の場合には演算エラーとなり，エラーフラグ（SM0）が ON し，エラーコードが SD0 に格納されます。

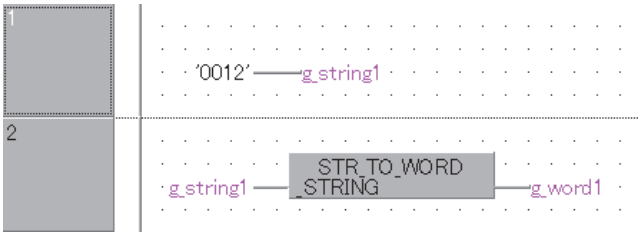
- ⓢの各桁のアスキーコードが，30H ～ 39H，41H ～ 46H 以外のとき。
  （エラーコード：4100）
- ⓢで指定されたデバイスが該当デバイスの範囲を超えるとき。
  （ユニバーサルモデル QCPU，LCPU の場合） （エラーコード：4101）

プログラム例

(1) ⓢに入力された文字列型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_WORD)

〔構造化ラダー/FBD〕

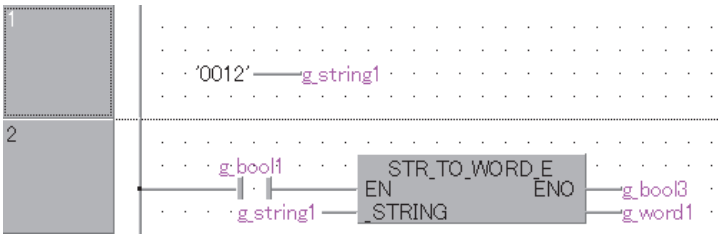


〔ST〕

```
g_string1 :="0012"
g_word1 := STR_TO_WORD (g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_WORD_E)

〔構造化ラダー/FBD〕



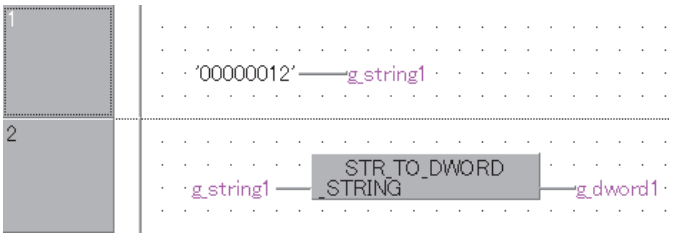
〔〔ST〕

```
g_string1 :="0012";
g_bool3 := STR_TO_WORD_E(g_bool1, g_string1, g_word1);
```

(2) ⓢに入力された文字列型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力プログラム。

(a) EN/ENO なし関数 (STR_TO_DWORD)

〔構造化ラダー/FBD〕



〔ST〕

```
g_string1 :="00000012";
g_dword1 := STR_TO_DWORD (g_string1);
```

5.1.31 文字列型→時間型変換

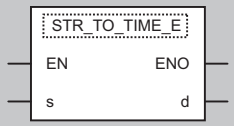
STR_TO_TIME(_E)

Basic High performance Process Redundant Universal LCPU

STR_TO_TIME(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= STR_TO_TIME_E (EN, s, d);

[ ] には，下記の関数が入ります。

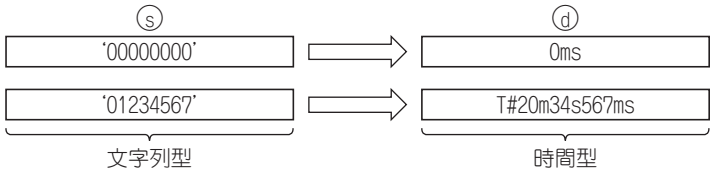
STR_TO_TIME　　STR_TO_ TIME _E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_STRING):	入力	:文字列(11)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:時間

機　能

演算処理

ⓢに入力された文字列型のデータを，時間型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
STR_TO_TIME(_E) : DDABIN

次の場合には演算エラーとなり，エラーフラグ（SM0）が ON し，エラーコードが SD0 に格納されます。

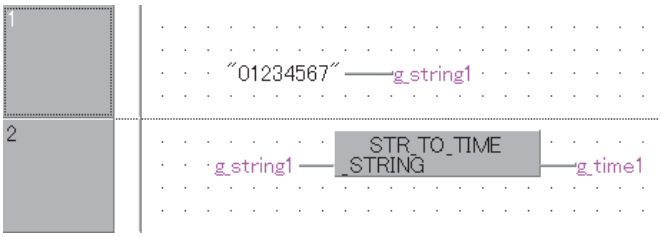
- ⓢの各桁のアスキーコードが，30H ～ 39H，20H，00H 以外のとき。（エラーコード：4100）
- ⓢのアスキーデータが下記範囲外のとき。（エラーコード：4100）
  -2147483648 ～ 4147483647

プログラム例

ⓢに入力された文字列型のデータを，時間型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_TIME)

〔構造化ラダー/FBD〕

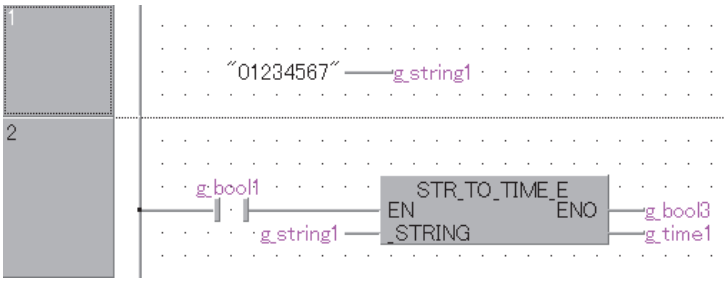


〔ST〕

```
g_string1 :="01234567";
g_time1 := STR_TO_TIME (g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_TIME_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_string1 :="01234567";
g_bool3 := STR_TO_TIME_E (g_bool1, g_string1, g_time1);
```

5.1.32 文字列型→ BCD 型変換

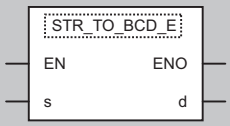
STR_TO_BCD(_E)

Basic High performance Process Redundant Universal LCPU

STR_TO_BCD(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= STR_TO_BCD_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

STR_TO_BCD　　STR_TO_ BCD _E

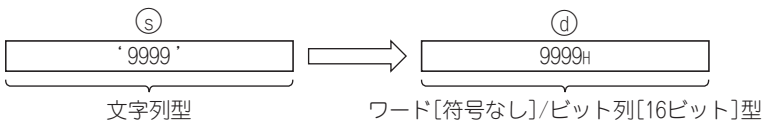
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_STRING):	入力	:文字列(8)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY_BIT

機　能

演算処理

(1) 出力引数にワード [符号なし]/ ビット列 [16 ビット] 型を指定した場合

(a) ⓢに入力された文字列型の 4 文字分のデータを，BCD 型のデータに変換してⓓから出力します。



(b) 文字列中に "20H"（スペース）が含まれていた場合は，"20H" を無視して変換します。

(c) 文字列中の "20H"（スペース），"30H" (0) も 1 文字としてカウントします。

(d) ⓢへの入力値は，文字列型のデータ値で下記に示す範囲内です。
アスキーコードで "30H" ～ "39H"，"20H"，"00H" の範囲。

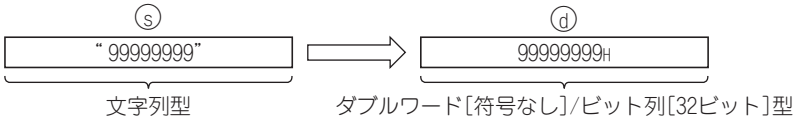
(e) 入力文字列が 4 文字に満たない場合は，文字列の後ろに 0 を補完した 4 文字で変換します。このため，4 文字に満たない文字列を BCD 変換する場合は，0 パディングした文字列（"1" の場合 "0001"）を入力してください。

(f) 文字列が 4 文字を越える場合は，左から 4 文字が変換対象となります。

入力文字列	変換文字列	出力 (BCD 型)
"1"	"1000"	1000H(4096)
"12"	"1200"	1200H(4608)
"123"	"1230"	1230H(4656)
"1234"	"1234"	1230H(4656)
"12345"	"1234"	1230H(4656)

(2) 出力引数にダブルワード [符号なし]/ ビット列 [32 ビット] 型を指定した場合

(a) ⓢに入力された文字列型の 8 文字分のデータを，BCD 型のデータに変換してⓓから出力します。



(b) 文字列中に "20H"（スペース）が含まれていた場合は，"20H" を無視して変換します。

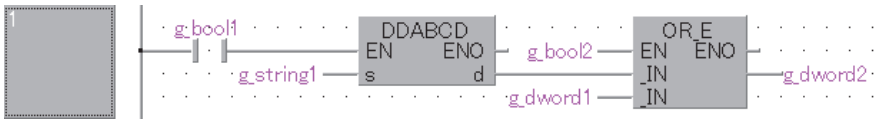
(c) 文字列中の "20H"（スペース），"30H" (0) も 1 文字としてカウントします。

(d) ⓢへの入力値は，文字列型のデータ値で下記に示す範囲内です。
アスキーコードで "30H" ～ "39H"，"20H"，"00H" の範囲。

(3) ⓓには，ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型が指定できます。ビット型は指定できません。

ポイント

ⓓからの出力は，ファンクションおよびオペレータのダブルワード [符号なし] / ビット列 [32 ビット] 型の入力に接続して使用できません。この場合は，DDABCD 命令を使用してください。



演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

下記の場合は演算エラーとなります。

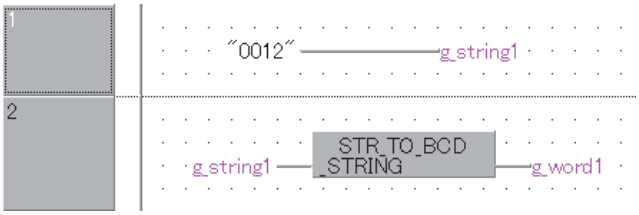
- アスキーコードで "30H" ～ "39H"，"20H"，"00H" 以外の範囲の文字列を入力した場合
（エラーコード 4100）

プログラム例

(1) ⓢに入力された文字列型のデータを，BCD 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_BCD)

〔構造化ラダー/FBD〕

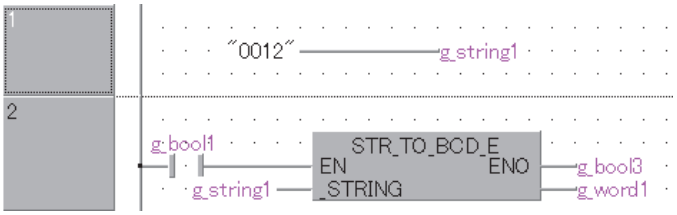


〔ST〕

```
g_string1:="0012";
g_word1 := STR_TO_BCD (g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_BCD_E)

〔構造化ラダー/FBD〕



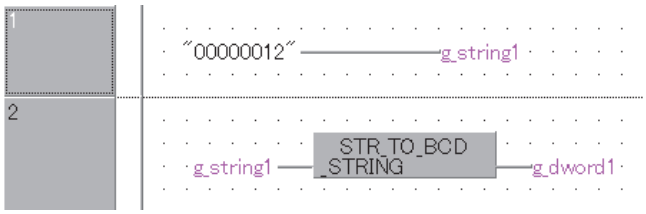
〔ST〕

```
g_string1:="0012";
g_bool3 := STR_TO_BCD_E (g_bool1, g_string1, g_word1);
```

(2) ⓢに入力された文字列型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型の BCD 型データに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (STR_TO_BCD)

〔構造化ラダー/FBD〕

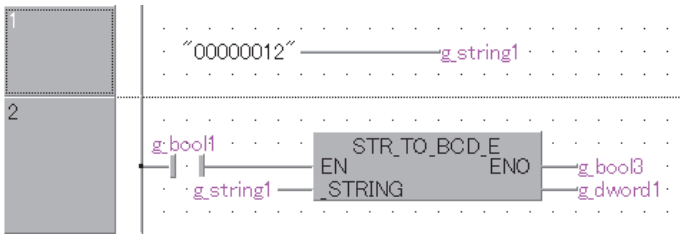


〔ST〕

```
g_string1:="00000012";
g_dword1:= STR_TO_BCD(g_string1);
```

(b) EN/ENO 付き関数 (STR_TO_BCD_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_string1:="00000012";
g_bool3:= STR_TO_BCD_E(g_bool1, g_string1, g_dword1);
```

5.1.33 BCD 型→ワード［符号付き］型，ダブルワード［符号付き］型変換

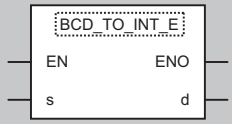
BCD_TO_INT(_E), BCD_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

BCD_TO_INT(_E)
BCD_TO_DINT(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD



ST

```
ENO:= BCD_TO_INT_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

BCD_TO_INT	BCD_TO_INT_E
BCD_TO_DINT	BCD_TO_DINT_E

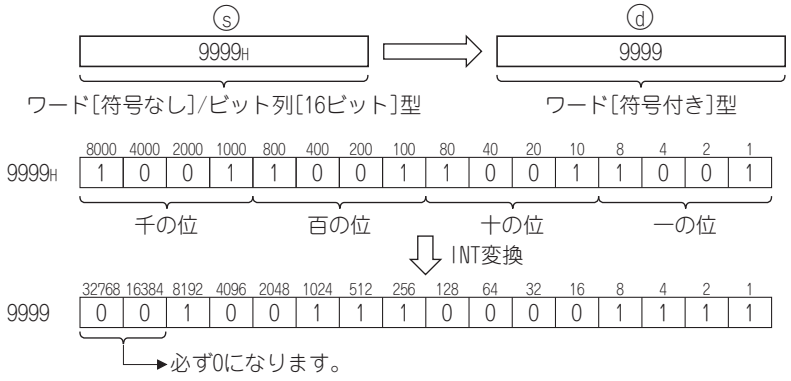
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s(_BCD):	入力	：ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	：ビット
	d:	出力	：ワード［符号付き］，ダブルワード［符号付き］

機　能

演算処理

(1) BCD_TO_INT, BCD_TO_INT_E

(a) ⓢに入力された BCD 型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。

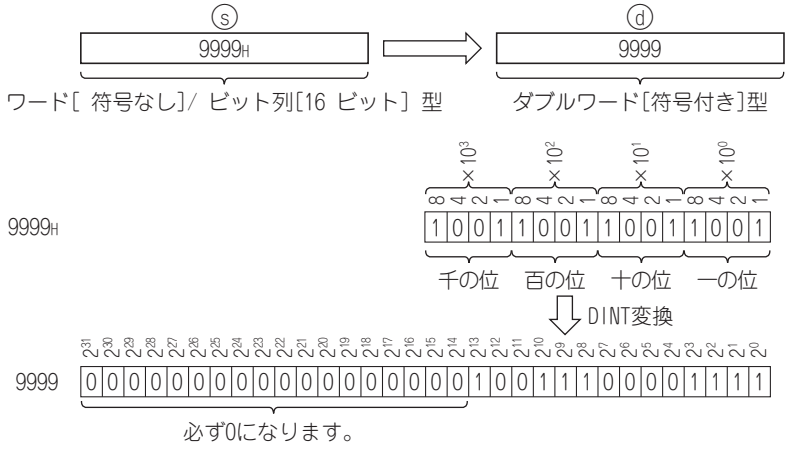


(b) ⓢへの入力値は，ワード［符号なし］/ ビット列［16 ビット］型のデータ値で 0H ～ 9999H（各桁は 0 ～ 9 の範囲）の範囲内です。

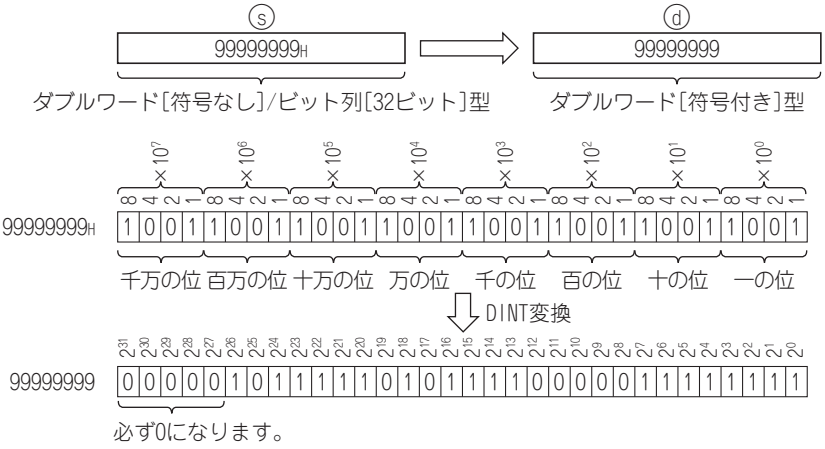
(2) BCD_TO_DINT, BCD_TO_DINT_E

(a) ⓢに入力された BCD 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力します。

• ⓢにワード [符号なし]/ ビット列 [16 ビット] を指定した場合



• ⓢにダブルワード [符号なし]/ ビット列 [32 ビット] を指定した場合



(b) ⓢへの入力値は，ワード [符号なし]/ ビット列 [16 ビット] 型データ値で 0H ～ 9999H（各桁は 0 ～ 9 の範囲），ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータ値で 0H ～ 99999999H（各桁は 0 ～ 9 の範囲）の範囲内です。

(c) ⓓには，ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型が指定できます。ビット型は指定できません。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
BCD_TO_INT(_E) : BIN
BCD_TO_DINT(_E) : BIN, WAND

次の場合には演算エラーとなり，エラーフラグ (SM0) が ON し，エラーコードが SD0 に格納されます。

- ⓢの各桁に 0 ～ 9 以外の値があるとき。 （エラーコード：4100）

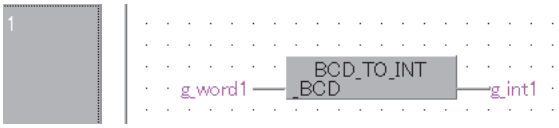
SM722 を ON しておくことにより，上記エラーを出さないようにすることができます。なお，範囲外の数値を設定した場合，SM722 の ON/OFF にかかわらず命令は実行されません。
また，BINP/DBINP 命令の場合，エラーの有無にかかわらず指令（実行条件）を OFF → ON させるまで次の演算は実行されません。

プログラム例

(1) ⓢに入力された BCD 型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BCD_TO_INT)

〔構造化ラダー/FBD〕

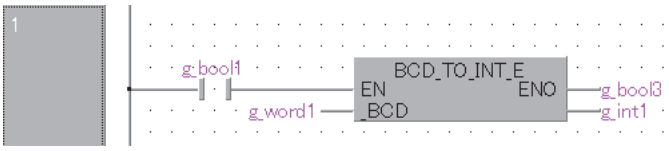


〔ST〕

```
g_int1 := BCD_TO_INT (g_word1);
```

(b) EN/ENO 付き関数 (BCD_TO_INT_E)

〔構造化ラダー/FBD〕



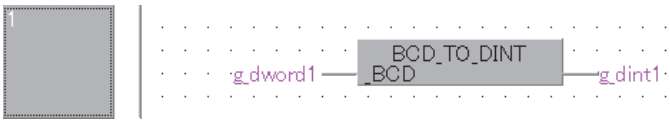
〔ST〕

```
g_bool3 := BCD_TO_INT_E (g_bool1, g_word1, g_int1);
```

(2) ⓢに入力された BCD 型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BCD_TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint1 := BCD_TO_DINT (g_dword1);
```

5.1.34 BCD 型→文字列型変換

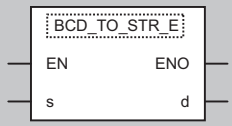
BCD_TO_STR(_E)

Basic High performance Process Redundant Universal LCPU

BCD_TO_STR(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= BCD_TO_STR_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

BCD_TO_STR　　BCD_TO_STR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_BCD):	入力	: ANY_BIT
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:文字列(8)

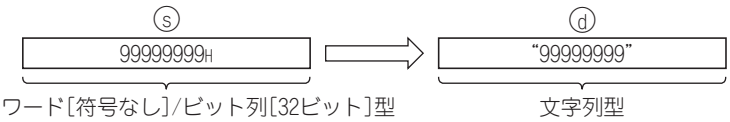
機　能

演算処理

(1) ⓢに入力された BCD 型のデータを，文字列型のデータに変換してⓓから出力します。

(a) ⓢにワード [符号なし]/ ビット列 [16 ビット] を指定した場合

(b) ⓢにダブルワード [符号なし]/ ビット列 [32 ビット] を指定した場合



(2) ⓢには，ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型が指定できます。ビット型は指定できません。

(3) SM701(出力文字数切換) が OFF の場合，文字列の最後に "00H" を格納します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
ⓢにワード [符号なし]/ ビット列 [16 ビット] 型を指定した場合 : BCDDA
ⓓにダブルワード [符号なし]/ ビット列 [32 ビット] 型を指定した場合 :DBCDDA

次の場合には演算エラーとなり，エラーフラグ (SM0) が ON し，エラーコードが SD0 に格納されます。

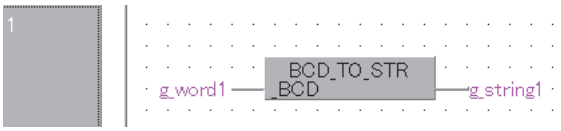
- ⓢにワード [符号なし]/ ビット列 [16 ビット] 型を指定した場合に，ⓢが0～9999以外のとき。（エラーコード：4100）
- ⓢにダブルワード [符号なし]/ ビット列 [32 ビット] 型を指定した場合に，ⓢが 0 ～ 99999999 以外のとき。（エラーコード：4100）
- ⓓで指定されたデバイスが該当デバイスの範囲を超えるとき。（ユニバーサルモデル QCPU，LCPU の場合 ）（エラーコード：4101）

プログラム例

(1) ⓢに入力されたワード [符号なし]/ ビット列 [16 ビット] 型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BCD _TO_STR)

〔構造化ラダー/FBD〕

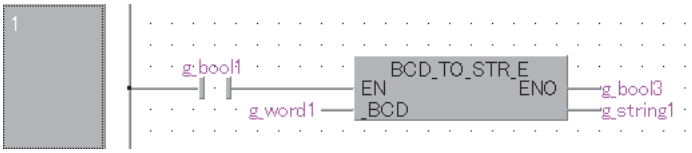


〔ST〕

```
g_string1 := BCD_TO_STR (g_word1);
```

(b) EN/ENO 付き関数 (BCD_TO_STR_E)

〔構造化ラダー/FBD〕



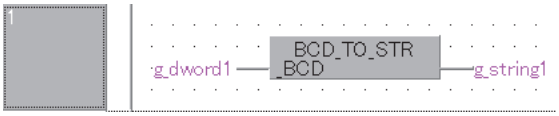
〔ST〕

```
g_bool3 := BCD_TO_STR_E (g_bool1, g_word1, g_string1);
```

(2) ⓢに入力されたダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BCD _TO_STR)

〔構造化ラダー/FBD〕

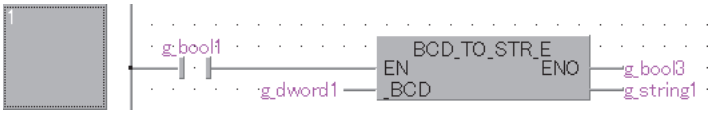


〔ST〕

```
g_string1 := BCD_TO_STR (g_dword1);
```

(b) EN/ENO 付き関数 (BCD_TO_STR_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := BCD_TO_STR_E (g_bool1, g_dword1, g_string1);
```

5.1.35 時間型→ビット型変換

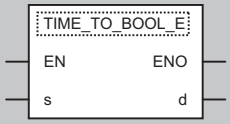
TIME_TO_BOOL(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

TIME_TO_BOOL(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= TIME_TO_BOOL_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

TIME_TO_BOOL　　TIME_TO_BOOL_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_TIME):	入力	:時間
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ビット

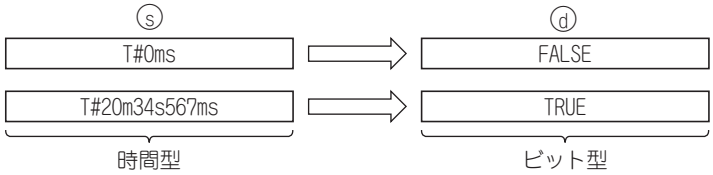
機　能

演算処理

ⓢに入力された時間型のデータを，ビット型のデータに変換してⓓから出力します。

入力値が 0ms の場合は，FALSE を出力します。

入力値が 0ms 以外の場合は，TRUE を出力します。



演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

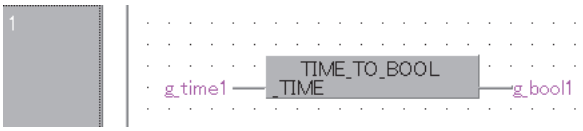
TIME_TO_BOOL(_E) での演算エラーはありません。

プログラム例

ⓢに入力された時間型のデータを，ビット型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (TIME_TO_BOOL)

〔構造化ラダー/FBD〕

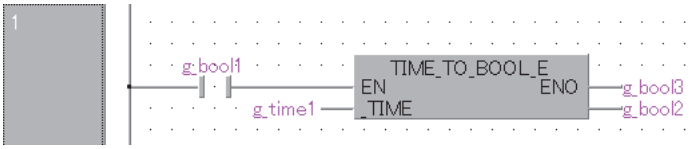


〔ST〕

```
g_bool1 := TIME_TO_BOOL (g_time1);
```

(b) EN/ENO 付き関数 (TIME_TO_BOOL_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := TIME_TO_BOOL_E (g_bool1, g_time1, g_bool2);
```

5.1.36 時間型→ワード［符号付き］型，ダブルワード［符号付き］型変換

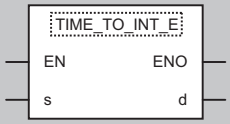
TIME_TO_INT(_E), TIME_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

TIME_TO_INT(_E)
TIME_TO_DINT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= TIME_TO_INT_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

TIME_TO_INT	TIME _TO_INT_E
TIME _TO_DINT	TIME _TO_DINT_E

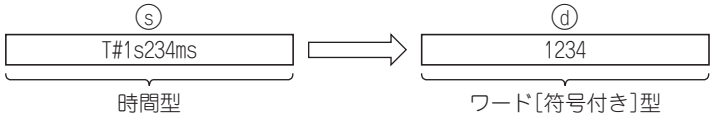
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_TIME):	入力	: 時間
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	: ビット
	d:	出力	: ワード［符号付き］，ダブルワード［符号付き］

機　能

演算処理

(1) TIME_TO_INT, TIME_TO_INT_E

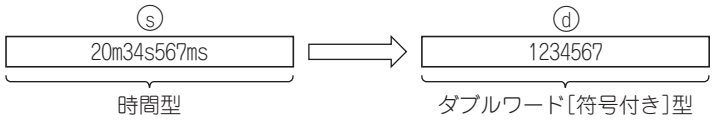
(a) ⓢに入力された時間型のデータを，ワード［符号付き］型のデータに変換してⓓから出力します。



(b) ワード［符号付き］型に変換時，時間型の上位 16 ビット（1 ワード分）のデータは切り捨てられます。

(2) TIME_TO_DINT, TIME_TO_DINT_E

ⓢに入力された時間型のデータを，ダブルワード［符号付き］型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

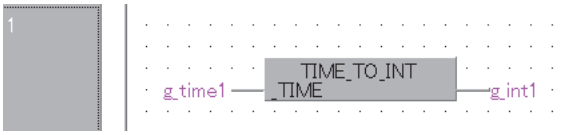
TIME_TO_INT(_E), TIME_TO_DINT(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力された時間型のデータを，ワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (TIME _TO_INT)

〔構造化ラダー/FBD〕

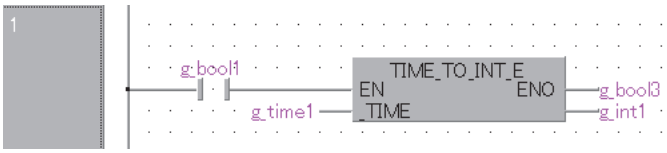


〔ST〕

```
g_int1 := TIME_TO_INT (g_time1);
```

(b) EN/ENO 付き関数 (TIME_TO_INT_E)

〔構造化ラダー/FBD〕



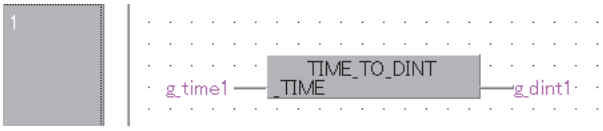
〔ST〕

```
g_bool3 := TIME_TO_INT_E (g_bool1, g_time1, g_int1);
```

(2) ⓢに入力された時間型のデータを，ダブルワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (TIME _TO_DINT)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint1 := TIME_TO_DINT (g_time1);
```

5.1.37 時間型→文字列型変換

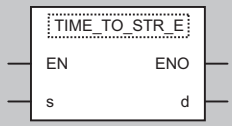
TIME_TO_STR(_E)

Basic High performance Process Redundant Universal LCPU

TIME_TO_STR(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= TIME_TO_STR_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

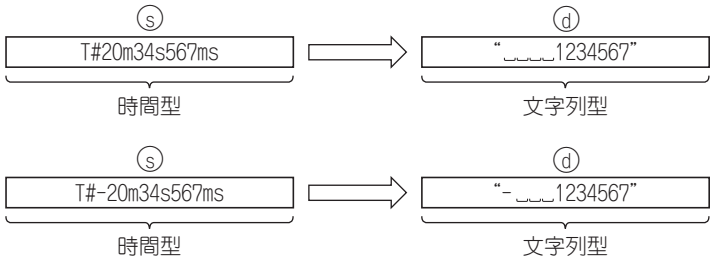
TIME_TO_STR　　TIME _TO_STR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_TIME):	入力	:時間
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:文字列(11)

機　能

演算処理

(1) ⓢに入力された時間型のデータを，文字列型のデータに変換してⓓから出力します。



(2) SM701（出力文字数切換）が OFF の場合，文字列の最後に "00H" を格納します。

(3) ⓓに格納する演算結果は，次のようになります。

(a) 1 文字目には，BIN データが正のとき 20H（スペース）を格納し，負のとき 2DH(-) を格納します。

(b) 有効桁数の左側には 20H（スペース）を格納します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
TIME_TO_STR(_E):DBINDA

次の場合には演算エラーとなり，エラーフラグ (SM0) が ON し，エラーコードが SD0 に格納されます。

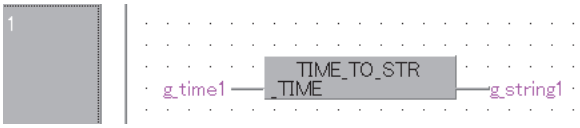
- ⓓに指定されたデバイスが該当デバイスの範囲を超えるとき。
（ユニバーサルモデル QCPU，LCPU の場合 ）　（エラーコード：4101）

プログラム例

ⓢに入力された時間型のデータを，文字列型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (TIME_TO_STR)

〔構造化ラダー/FBD〕

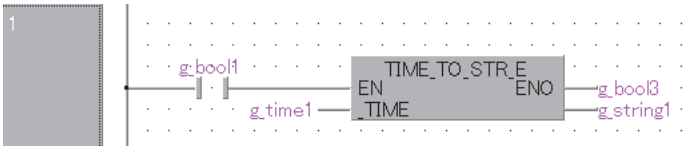


〔ST〕

```
g_string1 := TIME_TO_STR (g_time1);
```

(b) EN/ENO 付き関数例 (TIME_TO_STR_E)

〔構造化ラダー/FBD〕



〔ST〕

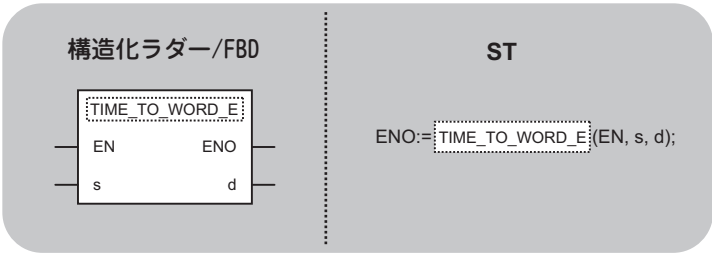
```
g_bool3 := TIME_TO_STR_E (g_bool1, g_time1, g_string1);
```

5.1.38 時間型→ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号なし］/ ビット列［32 ビット］型変換 TIME_TO_WORD(_E), TIME_TO_DWORD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

TIME_TO_WORD(_E)
TIME_TO_DWORD(_E)

（_E : EN/ENO 付き）



TIME_TO_WORD　TIME _TO_WORD_E
TIME _TO_DWORD　TIME _TO_DWORD_E
には，下記の関数が入ります。

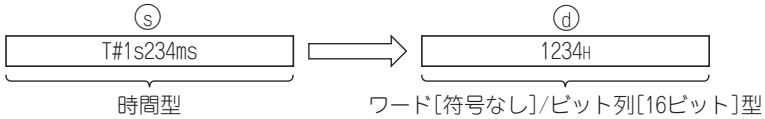
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_TIME):	入力	: 時間
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ワード［符号なし］/ ビット列［16 ビット］，ダブルワード［符号なし］/ ビット列［32 ビット］

機　能

演算処理

(1) TIME_TO_WORD, TIME_TO_WORD_E

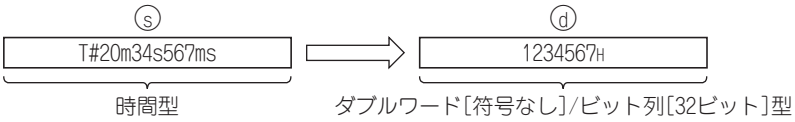
ⓢに入力された時間型のデータを，ワード［符号なし］/ ビット列［16 ビット］型のデータに変換してⓓから出力します。



(2) ワード［符号なし］/ ビット列［16 ビット］型に変換時，時間型の上位 16 ビット（1 ワード分）のデータは切り捨てられます。

(3) TIME_TO_DWORD, TIME_TO_DWORD_E

ⓢに入力された時間型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力します。



演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

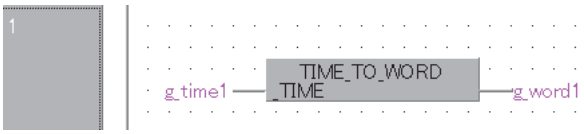
TIME_TO_WORD(_E), TIME_TO_DWORD(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力された時間型のデータを，ワード [符号なし]/ ビット列 [16 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (TIME _TO_WORD)

〔構造化ラダー/FBD〕

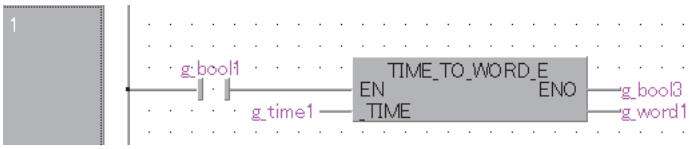


〔ST〕

```
g_word1 := TIME_TO_WORD (g_time1);
```

(b) EN/ENO 付き関数 (TIME_TO_WORD_E)

〔構造化ラダー/FBD〕



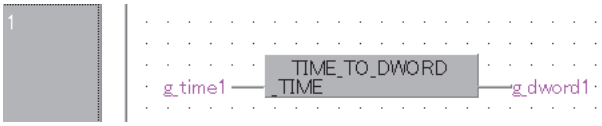
〔ST〕

```
g_bool3 := TIME_TO_WORD_E (g_bool1, g_time1, g_word1);
```

(2) ⓢに入力された時間型のデータを，ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (TIME _TO_DWORD)

〔構造化ラダー/FBD〕



〔ST〕

```
g_dword1 := TIME_TO_DWORD (g_time1);
```

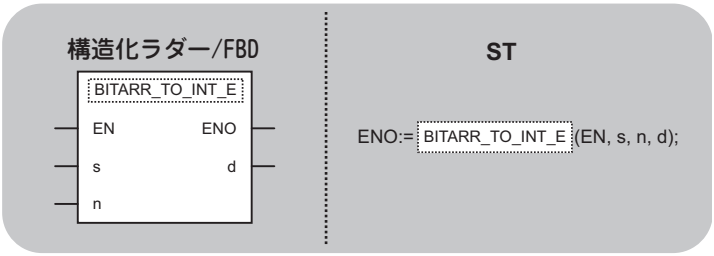
5.1.39 ビット配列→ワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号付き］型／ダブルワード［符号なし］/ ビット列［32 ビット］型変換

BITARR_TO_INT(_E), BITARR_TO_DINT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

BITARR_TO_INT(_E)
BITARR_TO_DINT(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

BITARR_TO_INT	BITARR_TO_INT_E
BITARR_TO_DINT	BITARR_TO_DINT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s(_BitArr):	入力（要素指定は変数も可）	：ビット
	n:	入力（4，8，12，16，20，24，28，32 の定数のみ指定可能）	：ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	：ビット
	d:	出力	：ANY16，ANY32

機　能

演算処理

(1) BITARR_TO_INT, BITARR_TO_INT_E

ⓢに入力されたビット配列要素を先頭として，n で指定されたビット数分をワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型のデータに変換してⓓから出力します。
n には，4，8，12，16 の定数のみ指定可能です。
指定されたビット数以上の出力ビットには 0 が設定されます。

(2) BITARR_TO_DINT, BITARR_TO_DINT_E

ⓢに入力されたビット配列要素を先頭として，n で指定されたビット数分をダブルワード［符号付き］型／ダブルワード［符号なし］/ ビット列［32 ビット］型のデータに変換してⓓから出力します。
n には，4，8，12，16，20，24，28，32 の定数のみ指定可能です。
指定されたビット数以上の出力ビットには 0 が設定されます。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

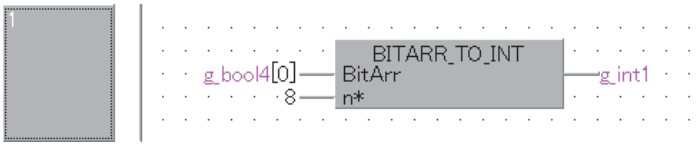
BITARR_TO_INT(_E), BITARR_TO_DINT(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット配列の 0 番目から 8 ビットをワード [符号付き] 型のデータに変換してⓓから出力するプログラム。

(a) EN/ENO なし関数 (BITARR_TO_INT)

〔構造化ラダー/FBD〕

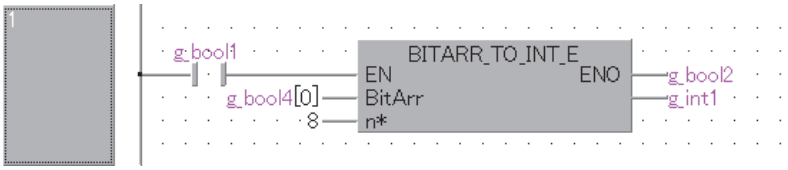


〔ST〕

```
g_int1 := BITARR_TO_INT(g_bool4[0], 8);
```

(b) EN/ENO 付き関数 (BITARR_TO_INT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool2 := BITARR_TO_INT_E(g_bool1, g_bool4[0], 8, g_int1);
```

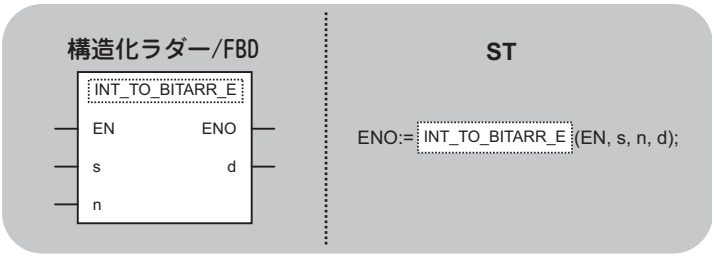
5.1.40 ワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型，ダブルワード［符号付き］型／ダブルワード［符号なし］/ ビット列［32 ビット］型→ビット配列変換

INT_TO_BITARR(_E), DINT_TO_BITARR(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

INT_TO_BITARR(_E)
DINT_TO_BITARR(_E)

（_E : EN/ENO 付き）



：には，下記の関数が入ります。

INT_TO_BITARR	INT_TO_BITARR_E
DINT_TO_BITARR	DINT_TO_BITARR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s:	入力	:ANY16，ANY32
	n:	入力（4，8，12，16，20，24，28，32 の定数のみ指定可能）	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d(_BitArr):	出力（要素指定は変数も可）	:ビット

機　能

演算処理

(1) INT_TO_BITARR, INT_TO_BITARR_E

ⓢに指定されたワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型の下位 n ビットをⓓに出力します。
n には，4，8，12，16 の定数のみ指定可能です。
指定されたビット数以上の出力ビットは変更しません。

(2) DINT_TO_BITARR, DINT_TO_BITARR_E

ⓢに指定されたダブルワード［符号付き］型／ダブルワード［符号なし］/ ビット列［32 ビット］型の下位 n ビットをⓓに出力します。
n には，4，8，12，16，20，24，28，32 の定数のみ指定可能です。
指定されたビット数以上の出力ビットは変更しません。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

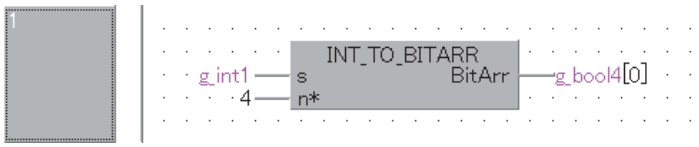
INT_TO_BITARR(_E)，DINT_TO_BITARR(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたワード [符号付き] 型データの下位 4 ビットをⓓに出力するプログラム。

(a) EN/ENO なし関数 (INT_TO_BITARR)

〔構造化ラダー/FBD〕

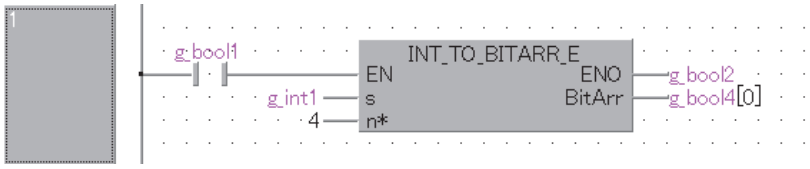


〔ST〕

```
g_bool4[0] := INT_TO_BITARR(g_int1, 4);
```

(b) EN/ENO 付き関数 (INT_TO_BITARR_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool2 := INT_TO_BITARR_E(g_bool1, g_int1, 4, g_bool4[0]);
```

5.1.41 ビット配列のコピー

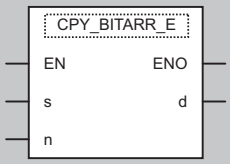
CPY_BITARR(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

CPY_BITARR(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= CPY_BITARR_E (EN, s, n, d);
```

[ ] には，下記の関数が入ります。

CPY_BITARR	CPY_BITARR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_BitArrIn):	入力	:ビット
	n:	入力（4，8，12，16，20，24，28，32 の定数のみ指定可）	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d(_BitArrOut):	出力	:ビット

機　能

演算処理

ⓢに入力されたビット配列の n ビットをⓓに出力します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

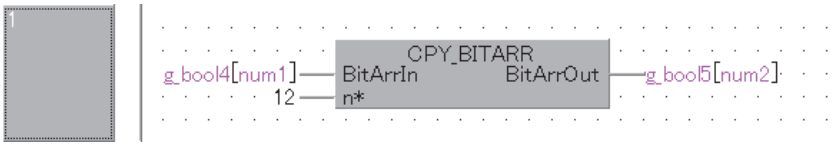
CPY_BITARR(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット配列の num1 番目の要素から 12 ビットをⓓの num2 番目以降に出力するプログラム。

(a) EN/ENO なし関数 (CPY_BITARR)

〔構造化ラダー/FBD〕

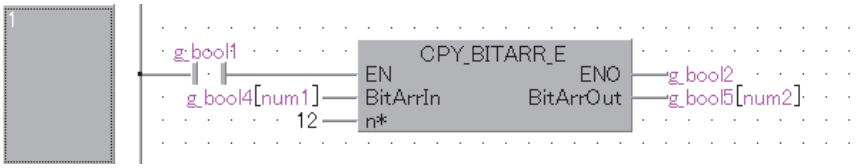


〔ST〕

```
g_bool5[num2] := CPY_BITARR(g_bool4[num1], 12);
```

(b) EN/ENO 付き関数 (CPY_BITARR_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool2 := CPY_BITARR_E(g_bool1, g_bool4[num1], 12, g_bool5[num2]);
```

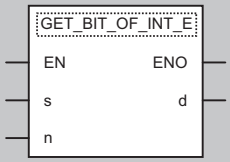
5.1.42 ワード［符号付き］型データの指定ビット読出し GET_BIT_OF_INT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

GET_BIT_OF_INT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= GET_BIT_OF_INT_E (EN, s, n, d);
```

[ ] には，下記の関数が入ります。

GET_BIT_OF_INT	GET_BIT_OF_INT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s:	入力	:ワード［符号付き］
	n:	入力（0 から 15 の定数のみ指定可能）	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d:	出力	:ビット

機　能

演算処理

ⓢの n ビット目の値を読み出し，ⓓから出力します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

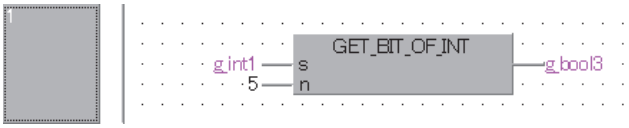
GET_BIT_OF_INT(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたデータの 5 ビット目を読み出し, ⓓから出力するプログラム。

(a) EN/ENO なし関数 (GET_BIT_OF_INT)

〔構造化ラダー/FBD〕

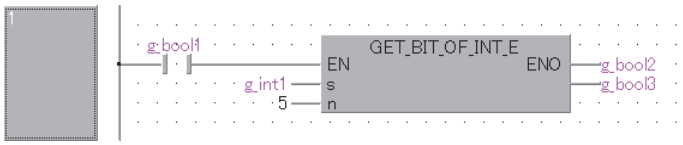


〔ST〕

```
g_bool3 := GET_BIT_OF_INT(g_int1, 5);
```

(b) EN/ENO 付き関数 (GET_BIT_OF_INT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool2 := GET_BIT_OF_INT_E(g_bool1, g_int1, 5, g_bool3);
```

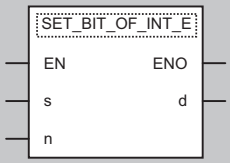
5.1.43 ワード［符号付き］型データの指定ビット書込み SET_BIT_OF_INT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

SET_BIT_OF_INT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= SET_BIT_OF_INT_E (EN, s, n, d);
```

[ ] には，下記の関数が入ります。

SET_BIT_OF_INT	SET_BIT_OF_INT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s:	入力	:ビット
	n:	入力（0 から 15 の定数のみ指定可能）	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d:	出力	:ワード［符号付き］

機　能

演算処理

ⓢで指定された値を，ⓓの n ビット目に書き込みます。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

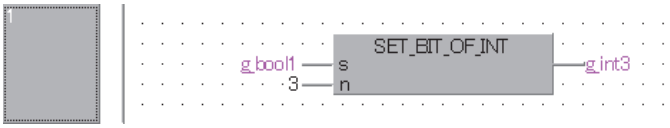
SET_BIT_OF_INT(_E) での演算エラーはありません。

プログラム例

ⓓの 3 ビット目に, ⓢの値を書き込むプログラム。

(a) EN/ENO なし関数 (SET_BIT_OF_INT)

〔構造化ラダー/FBD〕

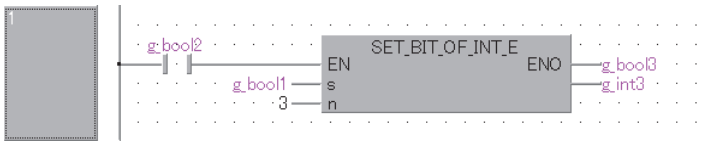


〔ST〕

```
g_int3 := SET_BIT_OF_INT(g_bool1. 3);
```

(b) EN/ENO 付き関数 (SET_BIT_OF_INT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := SET_BIT_OF_INT_E(g_bool2, g_bool1, 3, g_int3);
```

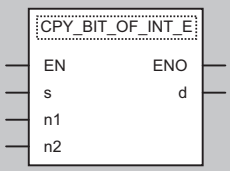
5.1.44 ワード［符号付き］型データの指定ビットコピー CPY_BIT_OF_INT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

CPY_BIT_OF_INT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= CPY_BIT_OF_INT_E (EN, s, n1, n2, d);
```

[ ] には，下記の関数が入ります。

CPY_BIT_OF_INT	CPY_BIT_OF_INT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s:	入力	:ワード［符号付き］
	n1:	入力（0 から 15 の定数のみ指定可能）	:ワード［符号付き］
	n2:	入力（0 から 15 の定数のみ指定可能）	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常実行　FALSE：異常または停止）	:ビット
	d:	出力	:ワード［符号付き］

機　能

演算処理

入力ⓢの n1 ビット目の値を，出力ⓓの n2 ビット目にコピーします。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

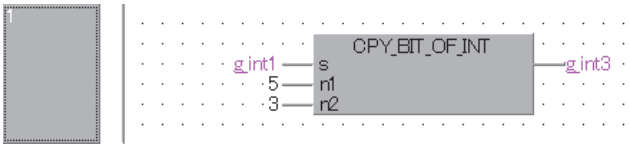
CPY_BIT_OF_INT(_E) での演算エラーはありません。

プログラム例

ⓢの 5 ビット目の値を，ⓓの 3 ビット目に書き込むプログラム。

(a) EN/ENO なし関数 (CPY_BIT_OF_INT)

〔構造化ラダー/FBD〕

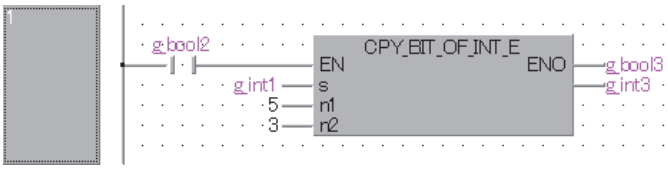


〔ST〕

```
g_int3 := CPY_BIT_OF_INT(g_int1, 5, 3);
```

(b) EN/ENO 付き関数 (CPY_BIT_OF_INT_E)

〔構造化ラダー/FBD〕



〔ST〕

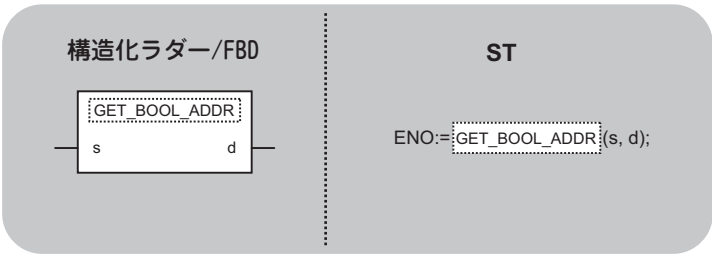
```
g_bool3 := CPY_BIT_OF_INT_E(g_bool2, g_int1, 5, 3, g_int3);
```

5.1.45 型変換の不要化

GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

Basic | High performance | Process | Redundant | Universal | LCPU

GET_BOOL_ADDR
GET_INT_ADDR
GET_WORD_ADDR



[ ] には，下記の関数が入ります。
GET_BOOL_ADDR
GET_INT_ADDR
GET_WORD_ADDR

入力引数，　s:　　入力
出力引数，　d:　　出力

: ANY
: ビット，ワード [符号付き]，
ワード [符号なし]/ ビット列 [16 ビット]

機　能

演算処理

(1) ⓢのデータ型をⓓのデータ型として出力します。

関数名	入力データ型	出力データ型
GET_BOOL_ADDR	ビット ビットの配列	ビット
GET_INT_ADDR	ワード [符号付き] ダブルワード [符号付き] ワード [符号なし]/ ビット列 [16 ビット] 単精度実数 文字列 時間型	ワード [符号付き]
GET_WORD_ADDR	ワード [符号付き] の配列 ダブルワード [符号付き] の配列 ワード [符号なし]/ ビット列 [16 ビット] の配列 ダブルワード [符号なし]/ ビット列 [32 ビット] の配列 実数の配列 時間型の配列	ワード [符号なし]/ ビット列 [16 ビット]

(2) プログラミングツールからⓢへの入力値を設定する場合，丸め誤差が発生する場合があります。プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

演算処理を実行します。ⓓからは，演算値が出力されます。

エラー

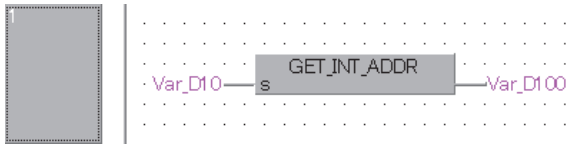
GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR での演算エラーは
ありません。

プログラム例

32 ビットの入力変数 Var_D10 を，型変換せずに直接 16 ビットの入力データとして扱う
プログラム。

GET_INT_ADDR

〔構造化ラダー/FBD〕



〔ST〕

```
Var_D100 := GET_INT_ADDR(Var_D10);
```

5.2 単数値変数関数

5.2.1 絶対値

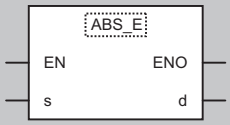
ABS(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

ABS(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= ABS_E (EN, s, d);
```

[ ] には，下記の関数が入ります。

ABS	ABS_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s(_IN):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ANY_NUM

機　能

演算処理

(1) ⓢに入力されたワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型データの絶対値を，ⓓからⓢと同一のデータ型で出力します。
入力値を A，演算出力値を B とすると，下記のようになります。

$$B = |A|$$

(2) ⓢへの入力値は，ワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型のデータ値です。

(3) ⓢのデータ型がワード [符号付き] 型で –32768 を入力した場合，–32768 がⓓから出力されます。

ⓢのデータ型がダブルワード [符号付き] 型で –2147483648 を入力した場合，

–2147483648 がⓓから出力されます。

（演算エラーにはなりません。また ABS_E のときは，ENO から TRUE が出力されます。）

(4) プログラミングツールからⓢに単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。

プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

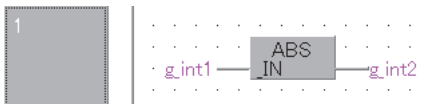
ABS(_E) での演算エラーはありません。

プログラム例

(1) ⓢに入力されたワード [符号付き] 型データの絶対値を，ⓓからⓢと同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (ABS)

〔構造化ラダー/FBD〕

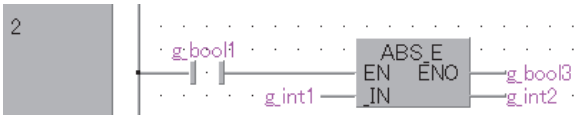


〔ST〕

```
g_int2:= ABS(g_int1);
```

(b) EN/ENO 付き関数 (ABS_E)

〔構造化ラダー/FBD〕



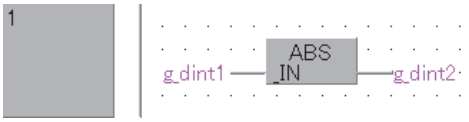
〔ST〕

```
g_bool3 := ABS_E(g_bool1, g_int1, g_int2);
```

(2) ⓢに入力されたダブルワード [符号付き] 型データの絶対値を，ⓓからⓢと同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (ABS)

〔構造化ラダー/FBD〕

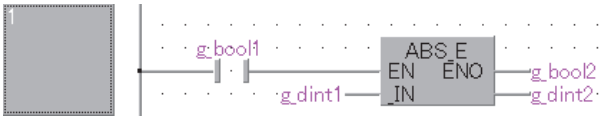


〔ST〕

```
g_dint2:= ABS(g_dint1);
```

(b) EN/ENO 付き関数 (ABS_E)

〔構造化ラダー/FBD〕



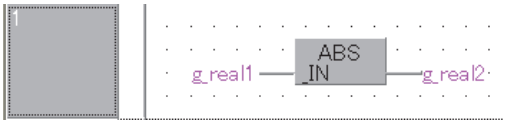
〔ST〕

```
g_bool2 := ABS_E(g_bool1, g_dint1, g_dint2);
```

(3) ⓢに入力された単精度実数型データの絶対値を，ⓓからⓢと同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (ABS)

〔構造化ラダー/FBD〕

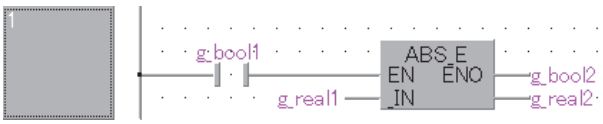


〔ST〕

```
g_real2:= ABS(g_real1);
```

(b) EN/ENO 付き関数 (ABS_E)

〔構造化ラダー/FBD〕



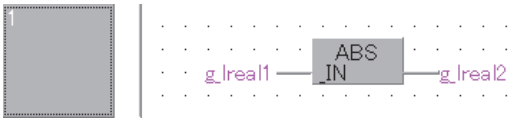
〔ST〕

```
g_bool2 := ABS_E(g_bool1, g_real1, g_real2);
```

(4) ⓢに入力された倍精度実数型データの絶対値を，ⓓからⓢと同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (ABS)

〔構造化ラダー/FBD〕



〔ST〕

```
g_lreal2:= ABS(g_lreal1);
```

(b) EN/ENO 付き関数 (ABS_E)

〔構造化ラダー/FBD〕

〔ST〕

```
g_bool2 := ABS_E(g_bool1, g_lreal1, g_lreal2);
```

5.3 算術演算関数

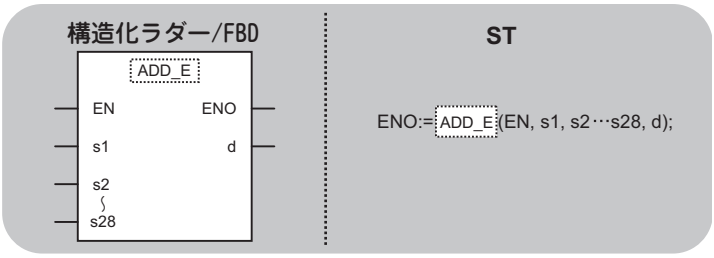
5.3.1 加算

ADD_E

Basic | High performance | Process | Redundant | Universal | LCPU

ADD_E

(_E : EN/ENO 付き)



ADD_E には，下記の関数が入ります。

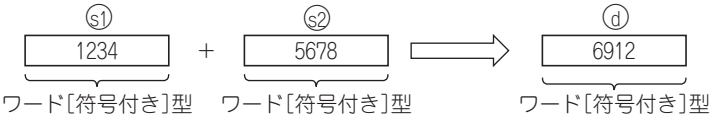
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	：ビット
	s1 ～ s28(_IN):	入力	：ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	：ビット
	d:	出力	：ANY_NUM

機　能

演算処理

(1) ⓢ1 ～ ⓢ28 に入力されたワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型データの加算（ⓢ1＋ⓢ2＋・・・＋ⓢ28）を行い，演算結果を ⓓ から ⓢ1 ～ ⓢ28 と同一のデータ型で出力します。

（例）データ型がワード [符号付き] 型の場合



(2) ⓢ1 ～ ⓢ28 への入力値は，ワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型のデータ値です。

(3) s のピン数は，2 ～ 28 の範囲で変更できます。

(4) 演算結果にオーバーフロー／アンダーフローが生じた場合は，ⓓから下記のように出力されます。

(a) データ型がワード [符号付き] 型の場合
オーバーフロー／アンダーフローが生じた場合でも，演算エラーにはなりません。
また，ADD_E の場合，ENO から TRUE が出力されます。
32767 + 2 = –32767 最上位ビットが 1 になるため，負の値になります。
(7FFFH) (0002H) (8001H)

–32767 + (–2) = 32766 最上位ビットが 0 になるため，正の値になります。
(8000H) (FFFEH) (7FFEH)

(b) データ型がダブルワード [符号付き] 型の場合
オーバーフロー／アンダーフローが生じた場合でも，演算エラーにはなりません。
また，ADD_E の場合，ENO から TRUE が出力されます。

2147483647 + 2 = –2147483647 最上位ビットが 1 になるため，負の値になります。
(7FFFFFFFH) (0002H) (80000001H)

–2147483648 + (–2) = 2147483646 最上位ビットが 0 になるため，正の値になります。
(80000000H) (FFFEH) (7FFFFFFEH)

(5) プログラミングツールからⓢ1～ⓢ28に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

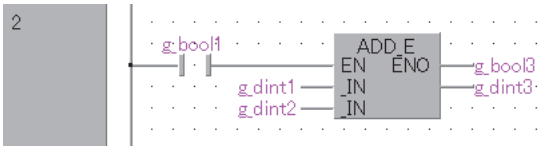
エラー

ADD_E での演算エラーはありません。

プログラム例

ⓢ1, ⓢ2 に入力されたダブルワード [符号付き] 型データの加算（ⓢ1+ⓢ2）を行い，演算結果をⓓから ⓢ1, ⓢ2 と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := ADD_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

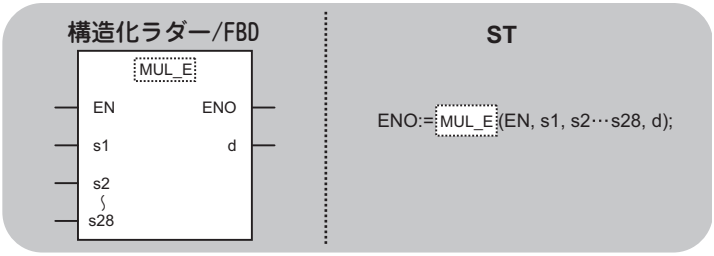
5.3.2 乗算

MUL_E

Basic | High performance | Process | Redundant | Universal | LCPU

MUL_E

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

MUL_E

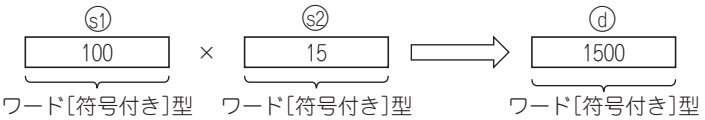
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1 ～ s28(_IN):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY_NUM

機　能

演算処理

(1) ⓢ1～ⓢ28に入力されたワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型データの乗算（ⓢ1×ⓢ2×・・・×ⓢ28）を行い，演算結果をⓓから ⓢ1～ⓢ28と同一のデータ型で出力します。

（例）データ型がワード [符号付き] 型の場合



(2) ⓢ1～ⓢ28への入力値は，ワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型のデータ値です。

(3) s のピン数は，2 ～ 28 の範囲で変更できます。

(4) 演算結果にアンダーフロー／オーバーフローが生じた場合は，ⓓは下記のように出力されます。

(a) データ型がワード [符号付き] 型の場合
アンダーフロー／オーバーフローが生じた場合でも，演算エラーにはなりません。
また，MUL_E の場合，ENO から TRUE が出力されます。

演算結果がワード [符号付き] 型データの範囲を超えている場合でも，ワード [符号付き] 型のデータを出力します。
（演算結果は 32 ビットデータになりますが，出力は上位 16 ビットを削除したワード [符号付き] 型のデータで出力されます。）
演算結果がワード [符号付き] 型データの範囲を超えている場合は，INT_TO_DINT により入力値をダブルワード [符号付き] 型データへ変換後，演算を行ってください。

(b) データ型がダブルワード [符号付き] 型の場合
アンダーフロー／オーバーフローが生じた場合でも，演算エラーにはなりません。
また，MUL_E の場合，ENO から TRUE が出力されます。

演算結果がダブルワード [符号付き] 型データの範囲を超えている場合でも，ダブルワード [符号付き] 型のデータを出力します。
（演算結果は 64 ビットデータになりますが，出力は上位 32 ビットを削除したダブルワード [符号付き] 型のデータで出力されます。）
演算結果がダブルワード [符号付き] 型データの範囲を超えている場合は，DINT_TO_REAL により入力値を単精度実数型データへ変換後，演算を行ってください。

(5) プログラミングツールからⓢ1～ⓢ28に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

ポイント

演算結果がデータ型の範囲を超える場合は，入力値のデータ型を変換してから演算を実行してください。

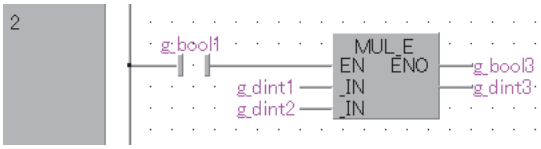
エラー

MUL_E での演算エラーはありません。

プログラム例

ⓢ1, ⓢ2に入力されたダブルワード [符号付き] 型データの乗算（ⓢ1×ⓢ2）を行い，演算結果をⓓから ⓢ1, ⓢ2と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := MUL_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

5.3.3 減算

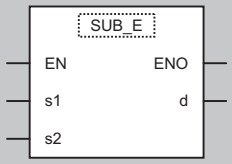
SUB_E

Basic | High performance | Process | Redundant | Universal | LCPU

SUB_E

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:=SUB_E(EN, s1, s2, d);
```

[ ] には，下記の関数が入ります。

SUB_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1): s2(_IN2):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY_NUM

機　能

演算処理

(1) ⓢ1, ⓢ2 に入力されたワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型データの減算（ⓢ1 − ⓢ2）を行い，演算結果をⓓから ⓢ1, ⓢ2 と同一のデータ型で出力します。

（例）データ型がワード [符号付き] 型の場合

(2) ⓢ1, ⓢ2 への入力値は，ワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型のデータ値です。

(3) 演算結果にアンダーフロー／オーバーフローが生じた場合は，ⓓは下記のように出力されます。

(a) データ型がワード [符号付き] 型の場合
アンダーフロー／オーバーフローが生じた場合でも，演算エラーにはなりません。
また，SUB_E の場合，ENO から TRUE が出力されます。
32767– （–2）= –32767　最上位ビットが 1 になるため，負の値になります。
(7FFFH) (FFFEH) (8001H)
–32767 – 2 = 32766　最上位ビットが 0 になるため，正の値になります。
(8000H) (0002H) (7FFEH)

(b) データ型がダブルワード [符号付き] 型の場合
アンダーフロー／オーバーフローが生じた場合でも，演算エラーにはなりません。
また，SUB_E の場合，ENO から TRUE が出力されます。

2147483647 – （–2）= –2147483647　最上位ビットが 1 になるため，負の値になります。
(7FFFFFFFH) (FFFEH) (80000001H)

–2147483648 – 2 = 2147483646　最上位ビットが 0 になるため，正の値になります。
(80000000H) (0002H) (7FFFFFFEH)

(4) プログラミングツールからⓢ1，ⓢ2に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

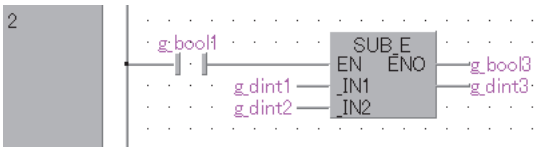
エラー

SUB_E での演算エラーはありません。

プログラム例

ⓢ1, ⓢ2 に入力されたダブルワード [符号付き] 型データの減算（ⓢ1−ⓢ2）を行い，演算結果をⓓからⓢ1, ⓢ2 と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := SUB_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

5.3.4 除算

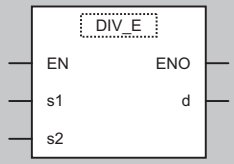
DIV_E

Basic | High performance | Process | Redundant | Universal | LCPU

DIV_E

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= DIV_E (EN, s1, s2, d);
```

[ ] には，下記の関数が入ります。

DIV_E

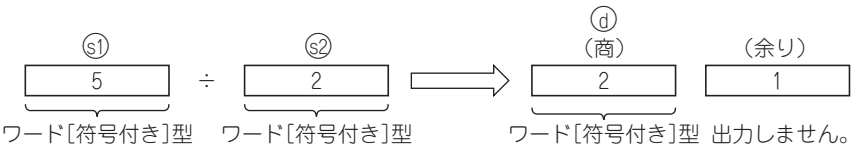
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1): s2(_IN2):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY_NUM

機　能

演算処理

(1) ⓢ1，ⓢ2に入力されたワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型データの除算（ⓢ1 ÷ ⓢ2）を行い，演算結果の商をⓓからⓢ1，ⓢ2と同一のデータ型で出力します。

（例）データ型がワード [符号付き] 型の場合



(2) ⓢ1，ⓢ2への入力値は，ワード [符号付き] 型／ダブルワード [符号付き] 型／単精度実数型／倍精度実数型のデータ値です。（ただし，ⓢ2へ入力される値は，0 以外です。）

(3) プログラミングツールからⓢ1，ⓢ2に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。

プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

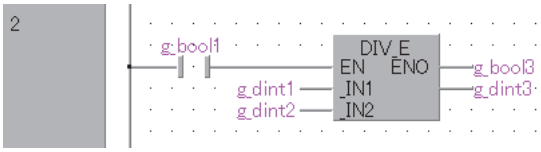
下記の場合には演算エラーとなります。

- ⓢ2 へ入力される値が 0 のとき。（0 割り）　（エラーコード 4100）

プログラム例

ⓢ1，ⓢ2 に入力されたダブルワード [符号付き] 型データの除算（ⓢ1 ÷ ⓢ2）を行い，演算結果の商をⓓから ⓢ1，ⓢ2 と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := DIV_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

5.3.5 剰余

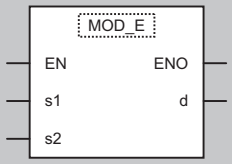
MOD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

MOD(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:=MOD_E(EN, s1, s2, d);
```

[ ] には，下記の関数が入ります。

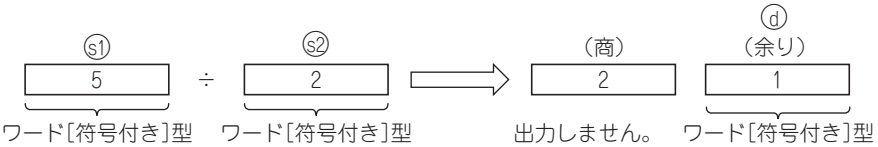
MOD	MOD_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	: ANY_INT
	s2(_IN2):		
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY_INT

機　能

演算処理

(1) ⓢ1, ⓢ2 に入力されたワード [符号付き] 型／ダブルワード [符号付き] 型データの除算（ⓢ1 ÷ ⓢ2）を行い，演算結果の余りをⓓからⓢ1, ⓢ2 と同一のデータ型で出力します。
（例）データ型がワード [符号付き] 型の場合



(2) ⓢ1, ⓢ2 への入力値は，ワード [符号付き] 型／ダブルワード [符号付き] 型のデータ値です。（ただし，ⓢ2 へ入力される値は，0 以外です。）

演算結果

(1) EN/ENO なし関数
演算結果を，下記に示します。

演算結果	ⓓ
演算エラーなし	演算出力値
演算エラーあり	不定値

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

下記の場合には演算エラーとなります。

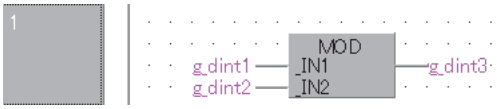
- ⓢ2へ入力される値が0のとき。（0割り）　（エラーコード 4100）

プログラム例

ⓢ1, ⓢ2に入力されたダブルワード [符号付き] 型データの除算（ⓢ1 ÷ ⓢ2）を行い，演算結果の余りをⓓからⓢ1, ⓢ2と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (MOD)

〔構造化ラダー/FBD〕

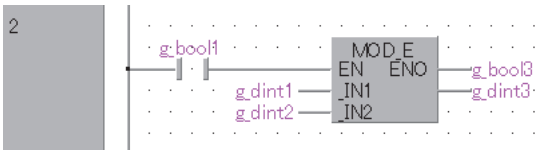


〔ST〕

```
g_dint3:= (g_dint1) MOD (g_dint2);
```

(b) EN/ENO 付き関数 (MOD_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := MOD_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

5.3.6 べき乗

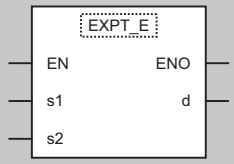
EXPT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

EXPT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:=EXPT_E(EN, s1, s2, d);
```

[ ] には，下記の関数が入ります。

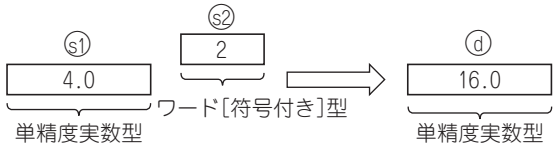
EXPT	EXPT _E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	: ANY_REAL
	s2(_IN2):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY_REAL

機　能

演算処理

(1) ⓢ1に入力された単精度実数型／倍精度実数型データを，ⓢ2で指定したワード［符号なし］／ダブルワード［符号なし］／単精度実数／倍精度実数型データでべき乗を行い，演算結果をⓓから出力します。



(2) プログラミングツールからⓢ1，ⓢ2に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
ⓢ1が単精度実数，ⓢ2がワード [符号付き] の場合 : LOG，FLT
ⓢ1が単精度実数，ⓢ2がダブルワード [符号付き] の場合 : LOG，DFLT
ⓢ1が単精度実数，ⓢ2が単精度実数 の場合 : LOG
ⓢ1が単精度実数，ⓢ2が倍精度実数 の場合 : LOGD，DFLTD
ⓢ1が倍精度実数，ⓢ2がワード [符号付き] の場合 : LOGD
ⓢ1が倍精度実数，ⓢ2がダブルワード [符号付き] の場合 : LOGD，FLTD
ⓢ1が倍精度実数，ⓢ2が単精度実数 の場合 : LOGD，DFLTD
ⓢ1が倍精度実数，ⓢ2が倍精度実数 の場合 : LOGD

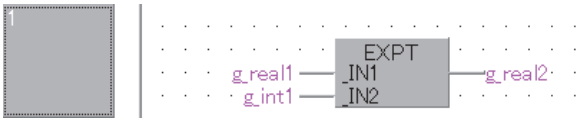
関数実行時に発生するエラーについては，MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編）を参照してください。

プログラム例

べき乗を行い，演算結果をⓓからⓢ1，ⓢ2と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (EXPT)

〔構造化ラダー/FBD〕

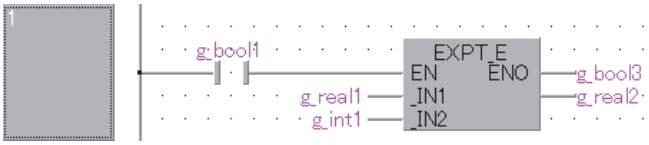


〔ST〕

```
g_real2:= EXPT(g_real1, g_int1);
```

(b) EN/ENO 付き関数 (EXPT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := EXPT_E(g_bool1, g_real1, g_int1, g_real2);
```

5.3.7 代入

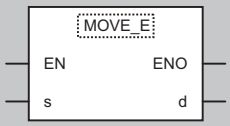
MOVE(_E)

Basic High performance Process Redundant Universal LCPU

MOVE(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= MOVE_E (EN, s, d);

[ ] には，下記の関数が入ります。

MOVE	MOVE_E

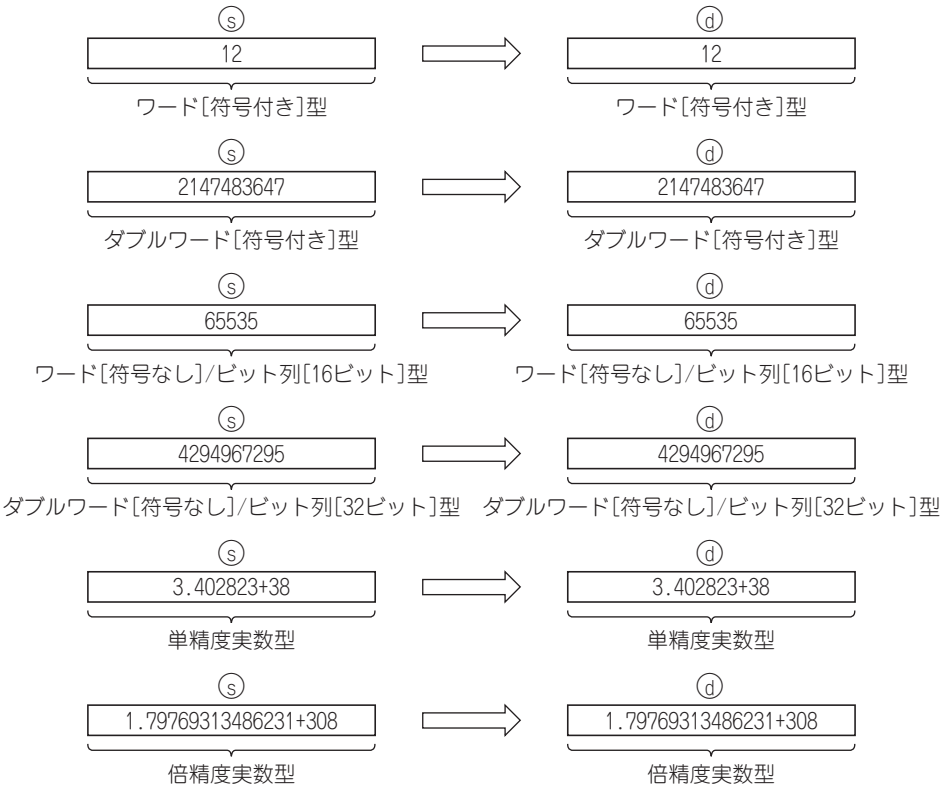
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	:ANY
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ANY

機　能

演算処理

(1) ⓢに入力されたデータを，ⓢと同一のデータ型でⓓへ代入します。

(2) ⓢ，ⓓには，ワード [符号付き] 型，ダブルワード [符号付き] 型，ワード [符号なし]/ ビット列 [16 ビット] 型，ダブルワード [符号なし]/ ビット列 [32 ビット] 型，単精度実数型，倍精度実数型，時間型，文字列型を指定することが可能です。またⓢ，ⓓとは同じデータ型のみ指定可能です。



(3) プログラミングツールからⓢに単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

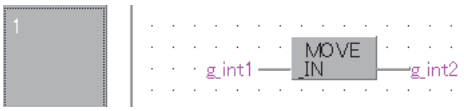
MOVE(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたワード [符号付き] 型データを，ⓓへ代入するプログラム。

(a) EN/ENO なし関数 (MOVE)

〔構造化ラダー/FBD〕

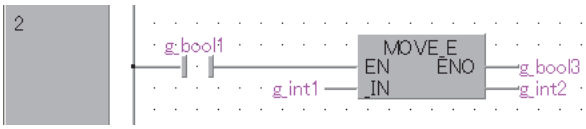


〔ST〕

```
g_int2:= MOVE(g_int1);
```

(b) EN/ENO 付き関数 (MOVE_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := MOVE_E(g_bool1, g_int1, g_int2);
```

5.4 ビット型ブール関数

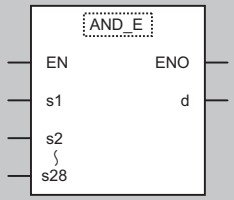
5.4.1 論理積，論理和，排他的論理和，論理否定 AND_E，OR_E，XOR_E，NOT(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

AND_E
OR_E
XOR_E
NOT(_E)

（ _E : EN/ENO 付き ）

構造化ラダー/FBD



ST

```
ENO:=AND_E(EN, s1, s2…s28, d);
```

[ ] には，下記の関数が入ります。

AND_E	
OR_E	
XOR_E	
NOT	NOT_E

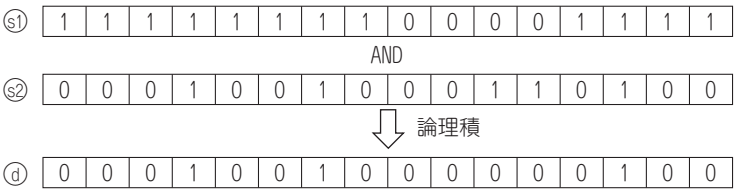
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s1 ～ s28(_IN1):	入力	: ANY_BIT
	(NOT(_E) は s1 のみ)		
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ANY_BIT

機　能

演算処理

(1) AND_E

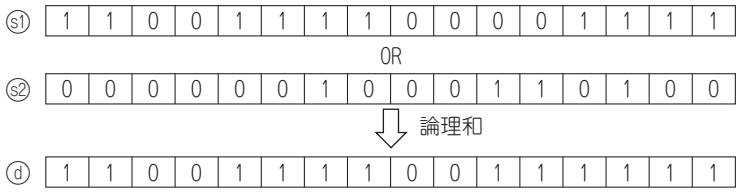
(a) ⓢ1～ⓢ28に入力されたビット型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型データをビットごとに論理積を行い，演算結果をⓓからⓢ1～ⓢ28と同一のデータ型で出力します。
（例）データ型がワード [符号なし]/ ビット列 [16 ビット] 型の場合



(b) s のピン数は，2 ～ 28 の範囲で変更できます。

(2) OR_E

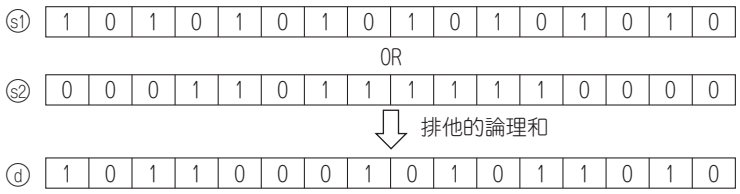
(a) ⓢ1～ⓢ28に入力されたビット型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型データをビットごとに論理和を行い，演算結果をⓓからⓢ1～ⓢ28と同一のデータ型で出力します。
(例) データ型がワード [符号なし]/ ビット列 [16 ビット] 型の場合



(b) s のピン数は，2 ～ 28 の範囲で変更できます。

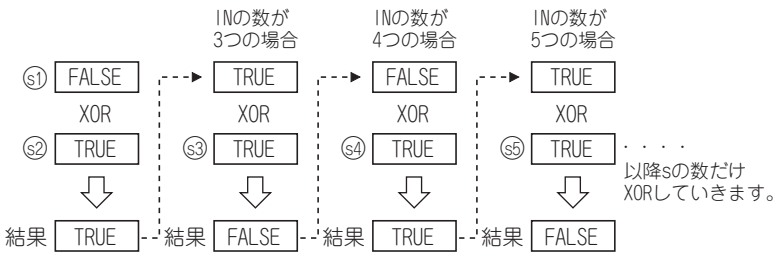
(3) XOR_E

(a) ⓢ1～ⓢ28に入力されたビット型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型データをビットごとに排他的論理和を行い，演算結果をⓓからⓢ1～ⓢ28と同一のデータ型で出力します。
(例) データ型がワード [符号なし]/ ビット列 [16 ビット] 型の場合



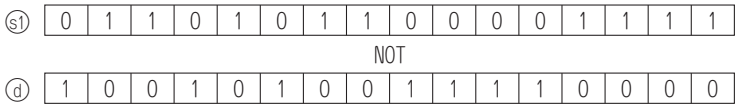
(b) s のピン数は，2 ～ 28 の範囲で変更できます。

(c) s が 3 つ以上ある場合，ⓢ1とⓢ2を XOR した結果に対して，ⓢ3を XOR します。
また，ⓢ4がある場合は，ⓢ3を XOR した結果に対してⓢ4を XOR します。
以降ⓢ5，ⓢ6と s の数だけ XOR していきます。
(例) データ型がビットの場合



(4) NOT, NOT_E

(a) ⓢ1に入力されたビット型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型データをビットごとに論理否定を行い，演算結果をⓓからⓢ1と同一のデータ型で出力します。
（例）データ型がワード [符号なし]/ ビット列 [16 ビット] 型の場合



(b) ⓢ1～ⓢ28への入力値は，ビット型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータ値です。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

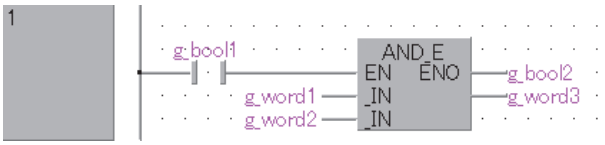
エラー

AND_E，OR_E，XOR_E，NOT(_E) での演算エラーはありません。

プログラム例

(1) ⓢ1，ⓢ2 に入力されたワード [符号なし]/ ビット列 [16 ビット] 型をビットごとに論理積を行い，演算結果を ⓓ から ⓢ1，ⓢ2 と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕

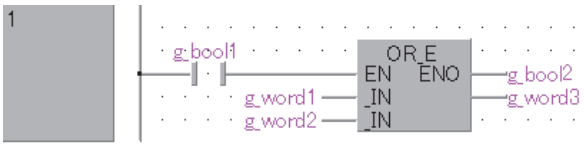


〔ST〕

```
g_bool2 := AND_E (g_bool1, g_word1, g_word2, g_word3)
```

(2) ⓢ1，ⓢ2 に入力されたワード [符号なし]/ ビット列 [16 ビット] 型をビットごとに論理和を行い，演算結果を ⓓ から ⓢ1，ⓢ2 と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕

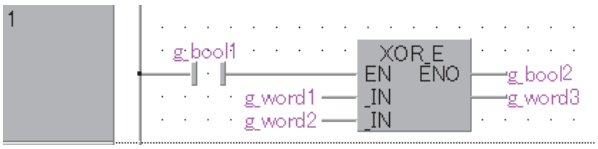


〔ST〕

```
g_bool2 := OR_E(g_bool1, g_word1, g_word2, g_word3);
```

(3) ⓢ1，ⓢ2 に入力されたワード [符号なし]/ ビット列 [16 ビット] 型をビットごとに排他的論理和を行い，演算結果を ⓓ から ⓢ1，ⓢ2 と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool2 := XOR_E(g_bool1, g_word1, g_word2, g_word3);
```

(4) ⓢ①に入力されたワード [符号なし]/ ビット列 [16 ビット] 型をビットごとに論理否定を行い, 演算結果をⓓからⓢ①と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (NOT)

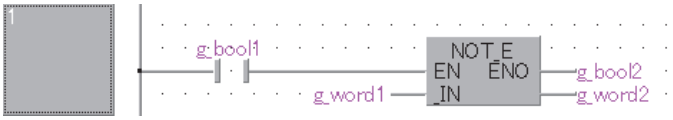
〔構造化ラダー/FBD〕

〔ST〕

```
g_word2 :=NOT (g_word1);
```

(b) EN/ENO 付き関数 (NOT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool2 := NOT_E (g_bool1, g_word1, g_word2);
```

5.5 選択関数

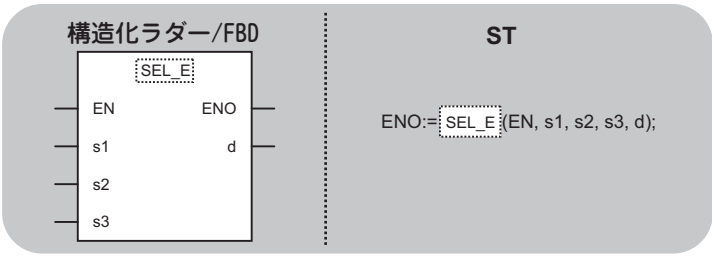
5.5.1 選択値

SEL(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

SEL(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

SEL	SEL_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s1(_G):	出力条件（TRUE：s3 出力　FALSE：s2 出力）	: ビット
	s2(_IN0):	入力	: ANY
	s3(_IN1):		
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: ANY

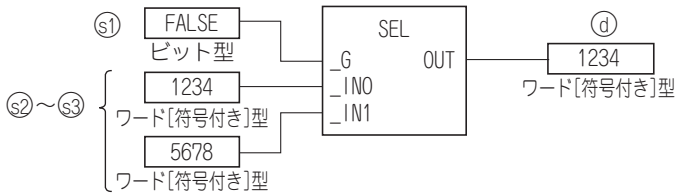
機　能

演算処理

(1) ⓢ1へ入力されたビット型データの入力値に従い，ⓢ2，ⓢ3へ入力された値のいずれかを，ⓓへⓢ2，ⓢ3と同一のデータ型で出力します。

ⓢ1の入力値が FALSE の場合，ⓢ2の入力値をⓓから出力します。
ⓢ1の入力値が TRUE の場合，ⓢ3の入力値をⓓから出力します。

(例) ⓢ2, ⓢ3 のデータ型が，ワード [符号付き] 型の場合



(2) ⓢ1 の入力値は，ビット型のデータ値です。

(3) ⓢ2, ⓢ3 への入力値は，ビット型／ワード［符号付き］型／ダブルワード［符号付き］型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型／単精度実数型／倍精度実数型／文字列型／時間型／構造体型／配列型のデータ値です。

(4) プログラミングツールから ⓢ2, ⓢ3 に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓ からは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓ から出力されるデータは不定値となります。その場合は，ⓓ から出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

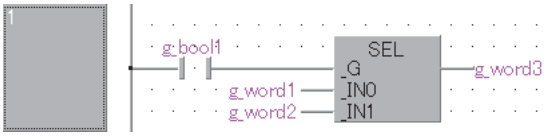
SEL(_E) での演算エラーはありません。

プログラム例

ⓢ1へ入力された入力値に従い，ⓢ2，ⓢ3へ入力された値のいずれかを，ⓓから ⓢ2，ⓢ3と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (SEL)

〔構造化ラダー/FBD〕

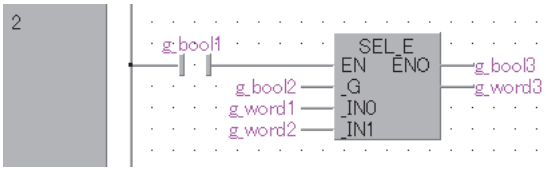


〔ST〕

```
g_word3 := SEL (g_bool1, g_word1, g_word2);
```

(b) EN/ENO 付き関数 (SEL_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := SEL_E (g_bool1, g_bool2, g_word1, g_word2, g_word3);
```

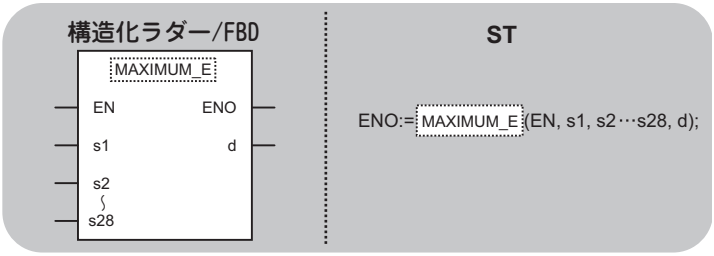
5.5.2 最大値，最小値選択

MAXIMUM(_E), MINIMUM(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

MAXIMUM(_E)
MINIMUM(_E)

(_E : EN/ENO 付き)



[ ] には，以下が入ります。

MAXIMUM	MAXIMUM_E
MINIMUM	MINIMUM_E

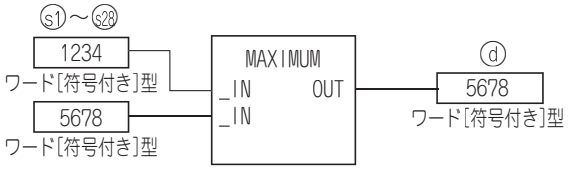
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1 ～ s28(_IN1)	入力	:ANY_SIMPLE
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ANY_SIMPLE

機　能

演算処理

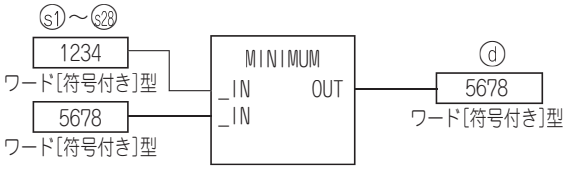
(1) MAXIMUM, MAXIMUM_E

ⓢ1～ⓢ28に入力されたビット型／ワード［符号付き］型／ダブルワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型／ダブルワード［符号なし］/ ビット列［32 ビット］型／単精度実数型／倍精度実数型／文字列型／時間型データの最大値を，ⓓから ⓢ1～ⓢ28と同一のデータ型で出力します。
（例）データ型がワード［符号付き］型の場合



(2) MINIMUM, MINIMUM_E

ⓢ1～ⓢ28に入力されたワード［符号付き］型／ダブルワード［符号付き］型／単精度実数型データの最小値を，ⓓから ⓢ1～ⓢ28と同一のデータ型で出力します。
（例）データ型がワード [符号付き] 型の場合



(3) ⓢ1～ⓢ28への入力値は，ビット型／ワード［符号付き］型／ダブルワード［符号付き］型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型／単精度実数型／倍精度実数型／文字列型／時間型のデータ値です。

(4) プログラミングツールからⓢ1～ⓢ28に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

(5) s のピン数は，2 ～ 28 の範囲で変更できます。

(6) ⓓにワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを指定した場合，C9026 のワーニングが発生します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

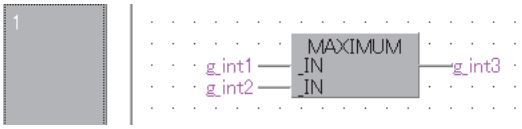
MAXIMUM(_E), MINIMUM(_E) での演算エラーはありません。

プログラム例

(1) ⓢ1, ⓢ2 に入力されたワード［符号付き］型データの最大値を, ⓓ から ⓢ1, ⓢ2 と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (MAXIMUM)

〔構造化ラダー/FBD〕

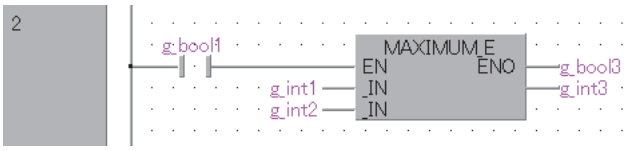


〔ST〕

```
g_int3 := MAXIMUM (g_int1, g_int2);
```

(b) EN/ENO 付き関数
(MAXIMUM_E)

〔構造化ラダー/FBD〕



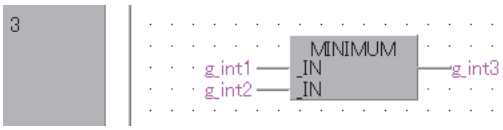
〔ST〕

```
g_bool3  := MAXIMUM_E (g_bool1, g_int1, g_int2, g_int3);
```

(2) ⓢ1, ⓢ2 に入力されたワード［符号付き］型データの最小値を, ⓓ から ⓢ1, ⓢ2 と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (MINIMUM)

〔構造化ラダー/FBD〕



〔ST〕

```
g_int3 := MINIMUM (g_int1, g_int2);
```

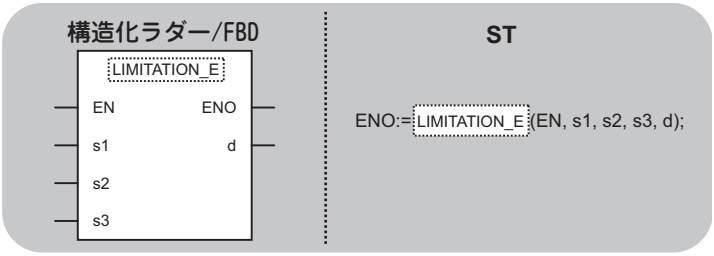
5.5.3 上下限リミット制御

LIMITATION(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

LIMITATION(_E)

(_E : EN/ENO 付き)



[ ] には，以下が入ります。

LIMITATION	LIMITATION_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_MN):	下限リミット値（最小出力限界値）	:ANY_SIMPLE
	s2(_IN):	上下限リミット制御により制御する入力値	:ANY_SIMPLE
	s3(_MX):	上限リミット値（最大出力限界値）	:ANY_SIMPLE
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:ANY_SIMPLE

機　能

演算処理

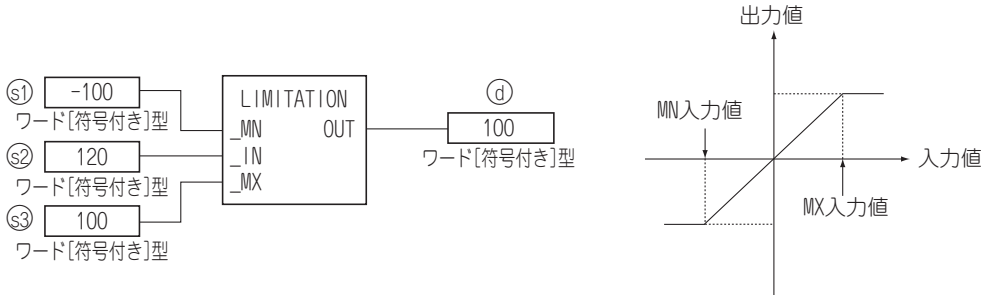
(1) ⓢ1, ⓢ2, ⓢ3へ入力されたビット型／ワード［符号付き］型／ダブルワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型／ダブルワード［符号なし］/ ビット列［32 ビット］型／単精度実数型／倍精度実数型／文字列型／時間型データに従い, ⓓから ⓢ1～ⓢ3と同一のデータ型で出力します。

(a) ⓢ2の入力値＞ⓢ3の入力値の場合, ⓢ3の入力値をⓓから出力します。

(b) ⓢ2の入力値＜ⓢ1の入力値の場合, ⓢ1の入力値をⓓからを出力します。

(c) ⓢ1の入力値≦ⓢ2の入力値≦ⓢ3の入力値の場合, ⓢ2の入力値をⓓから出力します。

（例）データ型がワード [符号付き] 型の場合



(2) ⓢ1, ⓢ2, ⓢ3への入力値は，ビット型／ワード［符号付き］型／ダブルワード［符号付き］型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型／単精度実数型／倍精度実数型／文字列型／時間型のデータ値です。（ただし，ⓢ1の入力値＜ⓢ3の入力値）

(3) プログラミングツールからⓢ1, ⓢ2, ⓢ3に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

(4) ⓓにワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型のデータを指定した場合，C9026 のワーニングが発生します。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

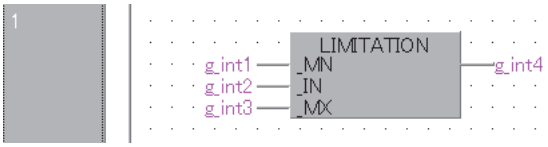
LIMITATION(_E) での演算エラーはありません。

プログラム例

(1) ⓢ1, ⓢ2, ⓢ3 へ入力されたワード［符号付き］型データに従い, ⓓ から ⓢ1, ⓢ2, ⓢ3 と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (LIMITATION)

〔構造化ラダー/FBD〕

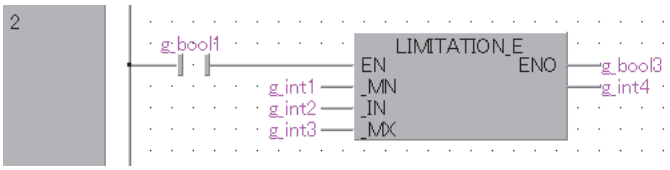


〔ST〕

```
g_int4 := LIMITATION (g_int1, g_int2, g_int3);
```

(b) EN/ENO 付き関数 (LIMITATION_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := LIMITATION_E (g_bool1, g_int1, g_int2, g_int3, g_int4);
```

5.5.4 マルチプレクサ

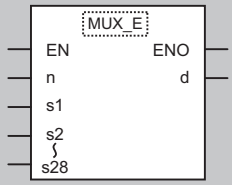
MUX(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

MUX(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:=MUX_E(EN, n, s1, s2…s28,d);
```

[ ] には，下記の関数が入ります。

MUX	MUX_E

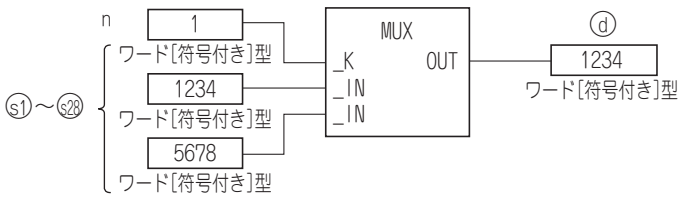
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	n(_K):	出力値選択	:ワード［符号付き］
	s1 ～ s28(_IN):	入力	: ANY
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	: ANY

機　能

演算処理

(1) n へ入力された入力値に従い，ⓢ1～ⓢ28へ入力された値のいずれかを，ⓓ から ⓢ1～ⓢ28と同一のデータ型で出力します。

n の入力値が 1 の場合，ⓢ1へ入力された値を ⓓ から出力します。
n の入力値が n の場合，ⓢn へ入力された値を ⓓ から出力します。
（例）データ型がワード [符号付き] 型の場合



(2) n に s のピン数の範囲外を入力した場合，ⓓ から不定値が出力されます。
（演算エラーにはなりません。また MUX_E のときは，ENO から FALSE が出力されます。）

(3) n への入力値は，ワード [符号付き] 型のデータ値で 1 ～ 28 の範囲内です。（ただし，s のピン数の範囲内です。）

(4) ⓢ1～ⓢ28への入力値は，ビット型／ワード [符号付き] 型／ダブルワード [符号付き] 型／ワード [符号なし]/ ビット列 [16 ビット] 型／ダブルワード [符号なし]/ ビット列 [32 ビット] 型／単精度実数型／倍精度実数型／文字列型／時間型／構造体型／配列型のデータ値です。

(5) プログラミングツールからⓢ1～ⓢ28に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

(6) s のピン数は，2 ～ 28 の範囲で変更できます。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

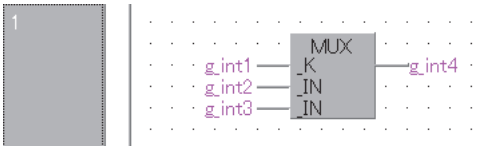
MUX(_E) での演算エラーはありません。

プログラム例

n へ入力された入力値に従い, ⓢ1～ⓢ2 へ入力された値のいずれかを, ⓓ から ⓢ1～ⓢ2 と同一のデータ型で出力するプログラム。

(a) EN/ENO なし関数 (MUX)

〔構造化ラダー/FBD〕

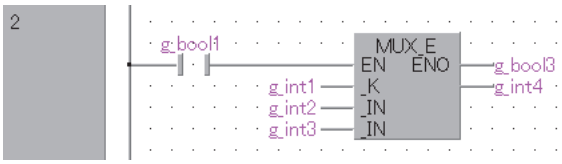


〔ST〕

```
g_int4 := MUX (g_int1, g_int2, g_int3);
```

(b) EN/ENO 付き関数 (MUX_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := MUX_E (g_bool1, g_int1, g_int2, g_int3, g_int4);
```

5.6 比較関数

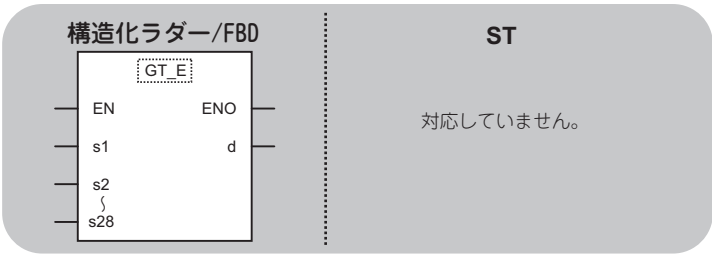
5.6.1 比較

GT_E, GE_E, EQ_E, LE_E, LT_E, NE_E

Basic | High performance | Process | Redundant | Universal | LCPU

GT_E
GE_E
EQ_E
LE_E
LT_E
NE_E

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。
GT_E
GE_E
EQ_E
LE_E
LT_E
NE_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s1 ～ s28(_IN):		: ANY_SIMPLE
	(NE_E は s1,s2 のみ)		
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力（TRUE：真値　FALSE：偽値）	: ビット

機　能

演算処理

(1) ⓢ1～ⓢ28に入力された入力値の比較演算を行い，演算結果をⓓからビット型で出力します。

(a) GT_E(>)[ⓢ1＞ⓢ2]&[ⓢ2＞ⓢ3]&・・・&[ⓢ(n−1) ＞ⓢ(n)] の比較を行います。

- すべてⓢ(n−1) ＞ⓢ(n) のとき，TRUE を出力します。
- いずれかがⓢ(n−1) ≦ⓢ(n) のとき，FALSE を出力します。

(b) GE_E(≧)[ⓢ1≧ⓢ2]&[ⓢ2≧ⓢ3]&・・・&[ⓢ(n−1) ≧ⓢ(n)] の比較を行います。

- すべてⓢ(n−1) ≧ⓢ(n) のとき，TRUE を出力します。
- いずれかがⓢ(n−1) ＜ⓢ(n) のとき，FALSE を出力します。

(c) EQ_E(＝)[ⓢ1＝ⓢ2]&[ⓢ2＝ⓢ3]&・・・&[ⓢ(n−1) ＝ⓢ(n)] の比較を行います。

- すべてⓢ(n−1) ＝ⓢ(n) のとき，TRUE を出力します。
- いずれかがⓢ(n−1) ≠ⓢ(n) のとき，FALSE を出力します。

(d) LE_E(≦)[ⓢ1≦ⓢ2]&[ⓢ2≦ⓢ3]&・・・&[ⓢ(n−1) ≦ⓢ(n)] の比較を行います。

- すべてⓢ(n−1) ≦ⓢ(n) のとき，TRUE を出力します。
- いずれかがⓢ(n−1) ＞ⓢ(n) のとき，FALSE を出力します。

(e) LT_E(<)[ⓢ1＜ⓢ2]&[ⓢ2＜ⓢ3]&・・・&[ⓢ(n−1) ＜ⓢ(n)] の比較を行います。

- すべてⓢ(n−1) ＜ⓢ(n) のとき，TRUE を出力します。
- いずれかがⓢ(n−1) ≧ⓢ(n) のとき，FALSE を出力します。

(f) NE_E(<>)[ⓢ1≠ⓢ2] の比較を行います。

- ⓢ1≠ⓢ2のとき TRUE を出力します。
- ⓢ1＝ⓢ2のとき FALSE を出力します。

(2) ⓢへの入力値は，ビット型／ワード［符号付き］型／ダブルワード［符号付き］型／ワード［符号なし］/ ビット列［16 ビット］型／ダブルワード［符号なし］/ ビット列［32 ビット］型／単精度実数型／倍精度実数型／文字列型／時間型のデータ値です。

(3) プログラミングツールからⓢ1～ⓢ28に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

(4) ⓢのピン数は，2 ～ 28 の範囲で変更できます。（ただし，NE_E の s のピンは，ⓢ1，ⓢ2固定です。）

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

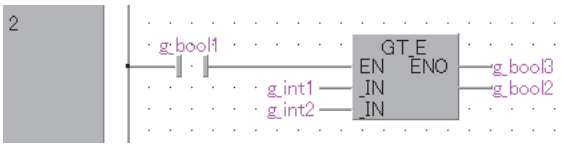
エラー

GT_E，GE_E，EQ_E，LE_E，LT_E，NE_E での演算エラーはありません。

プログラム例

ⓢ1，ⓢ2に入力された入力値の比較演算を行い，演算結果をⓓから出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := GT_E (g_bool1, g_int1, g_int2, g_bool2);
```

5.7 文字列関数

5.7.1 文字列の抽出

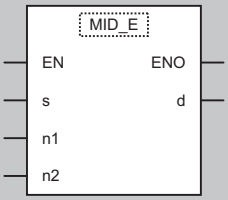
MID(_E)

Basic High performance Process Redundant Universal LCPU

MID(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

ENO:= MID_E (EN, s, n1, n2, d);

[ ] には，下記の関数が入ります。

MID　　MID_E

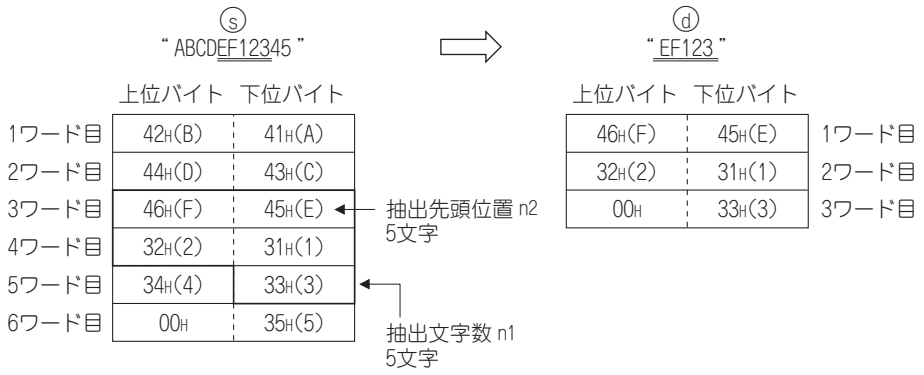
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_IN):	入力	:文字列(255)
	n1(_L):	抽出文字数指定	:ワード［符号付き］
	n2(_P):	抽出先頭位置指定	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:文字列(255)

機　能

演算処理

(1) ⓢへ入力された文字列の任意位置から指定文字数分のデータを，ⓓから出力します。
抽出する文字数は，n1 への入力値により指定します。
抽出する文字列の先頭位置は，n2 への入力値により指定します。

（例）n1，n2 への入力値が 5 の場合



(2) ⓢへの入力値は，文字列型のデータ値で 0 ～ 255 バイトの範囲内です。

(3) n1 への入力値は，ワード [符号付き] 型のデータ値で 0 ～ 255 の範囲内です。
（ただし，ⓢへ入力される文字列の文字数以内に限ります。）

(4) n2 への入力値は，ワード [符号付き] 型のデータ値で 1 ～ 255 の範囲内です。
（ただし，ⓢへ入力される文字列の文字数以内に限ります。）

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

本関数は，下記の共通命令を使用して作られています。
MID(_E): MIDR

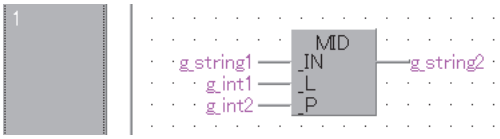
関数実行時に発生するエラーについては，MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編）を参照してください。

プログラム例

ⓢへ入力された文字列の任意位置から指定文字数分のデータを，ⓓから出力するプログラム。

(a) EN/ENO なし関数 (MID)

〔構造化ラダ ―/FBD〕

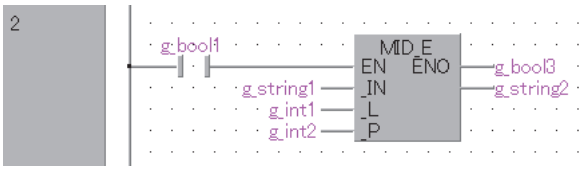


〔ST〕

```
g_string2:=MID(g_string1, g_int1, g_int2);
```

(b) EN/ENO 付き関数 (MID_E)

〔構造化ラダ ―/FBD〕



〔ST〕

```
g_bool3 := MID_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

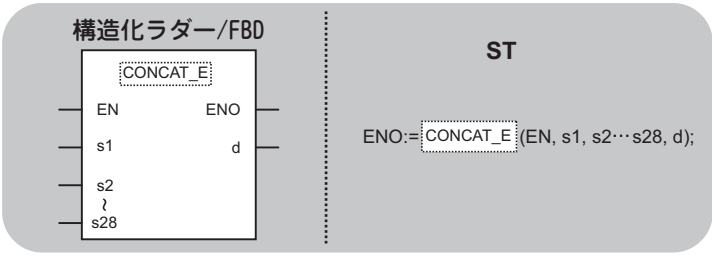
5.7.2 文字列の結合

CONCAT(_E)

Basic / High performance / Process / Redundant / Universal / LCPU

CONCAT(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

CONCAT	CONCAT_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1 ～ s28(_IN):	入力	:文字列 (255)
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:文字列 (255)

機　能

演算処理

(1) ⓢ1 へ入力された文字列のあとに，ⓢ2 ～ ⓢ28 へ入力された文字列を結合して，ⓓ から出力します。

文字列の結合は，ⓢ1 の文字列の終了を示す "00H" を無視して，ⓢ2 ～ ⓢ28 の文字列を続けて結合します。
結合した文字列が 255 バイトを超えている場合は，255 バイトまでの文字列を出力します。

ⓢ1 "ABCDE"

	上位バイト	下位バイト
1ワード目	42H(B)	41H(A)
2ワード目	44H(D)	43H(C)
3ワード目	00H	45H(E)

＋

ⓢ2 "123456"

	上位バイト	下位バイト
1ワード目	32H(2)	31H(1)
2ワード目	34H(4)	33H(3)
3ワード目	36H(6)	35H(5)
4ワード目	00H	

⇨ ⓓ "ABCDE123456"

	上位バイト	下位バイト
1ワード目	42H(B)	41H(A)
2ワード目	44H(D)	43H(C)
3ワード目	31H(1)	45H(E)
4ワード目	33H(3)	32H(2)
5ワード目	35H(5)	34H(4)
6ワード目	00H	36H(6)

(2) ⓢ1，ⓢ2 ～ ⓢ28 への入力値は，文字列型のデータ値で 0 ～ 255 バイトの範囲内です。

(3) s のピン数は，2 ～ 28 の範囲で変更できます。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

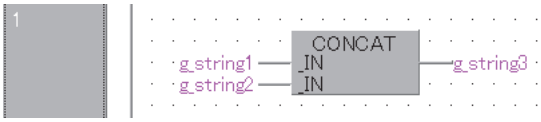
CONCAT(_E) での演算エラーはありません。

プログラム例

ⓢ1へ入力された文字列のあとに，ⓢ2へ入力された文字列を結合して，ⓓから出力するプログラム。

(a) EN/ENO なし関数 (CONCAT)

〔構造化ラダー/FBD〕

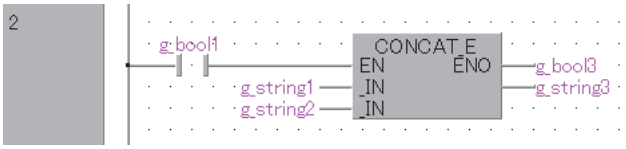


〔ST〕

```
g_string3:=CONCAT(g_string1, g_string2);
```

(b) EN/ENO 付き関数 (CONCAT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := CONCAT_E(g_bool1, g_string1, g_string2, g_string3);
```

5.7.3 文字列の挿入

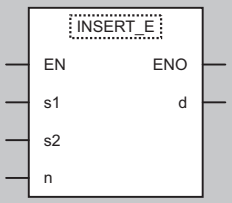
INSERT(_E)

Basic High performance Process Redundant Universal LCPU

INSERT(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= INSERT_E (EN, s1, s2, n, d);
```

[ ] には，下記の関数が入ります。

INSERT	INSERT_E

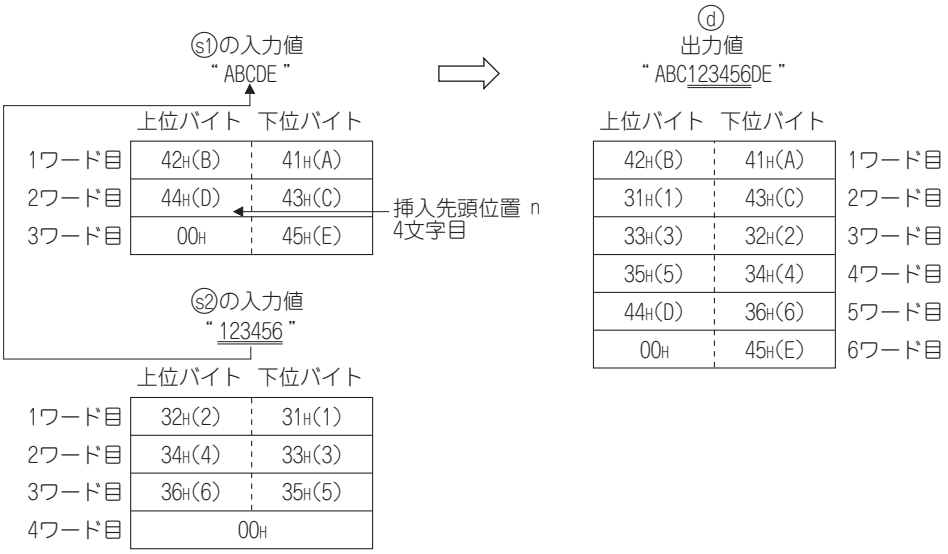
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s1(_IN1): s2(_IN2):	入力	: 文字列 (255)
	n(_P):	挿入先頭位置指定	: ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 文字列 (255)

機　能

演算処理

(1) ⓢ1 へ入力された文字列に，文字列の先頭から任意の位置へ ⓢ2 へ入力された文字列を挿入して，ⓓ から出力します。
挿入する文字列の先頭位置は，n への入力値により指定します。
ⓢ2 の文字列を ⓢ1 の文字列へ挿入後，ⓢ2 の文字列の終了を示す "00H" は無視します。挿入した文字列が 255 バイトを超えている場合は，255 バイトまでの文字列を出力します。

（例）n への入力値が，4 の場合



(2) ⓢ1，ⓢ2 への入力値は，文字列型のデータ値で 0 ～ 255 バイトの範囲内です。

(3) n への入力値は，ワード [符号付き] 型のデータ値で 1 ～ 255 の範囲内です。
（ただし，ⓢ1 へ入力される文字列の文字数以内に限ります。）

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓ からは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓ から出力されるデータは不定値となります。
その場合は，ⓓ から出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

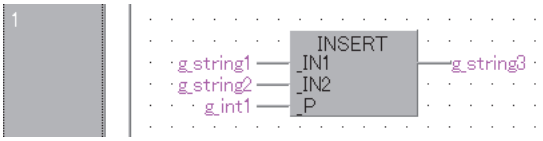
INSERT(_E) での演算エラーはありません。

プログラム例

ⓢ1へ入力された文字列に，文字列の先頭から任意の位置へⓢ2へ入力された文字列を挿入して，ⓓから出力するプログラム。

(a) EN/ENO なし関数 (INSERT)

〔構造化ラダー/FBD〕

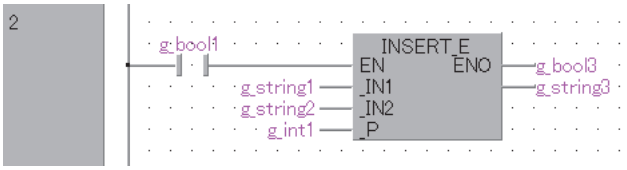


〔ST〕

```
g_string3:=INSERT(g_string1, g_string2, g_int1);
```

(b) EN/ENO 付き関数 (INSERT_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := INSERT_E(g_bool1, g_string1, g_string2, g_int1, g_string3);
```

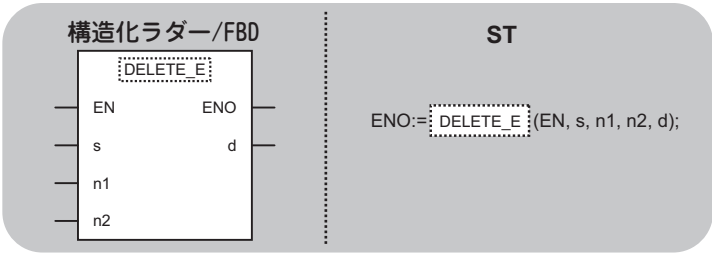
5.7.4 文字列の削除

DELETE(_E)

Basic / High performance / Process / Redundant / Universal / LCPU

DELETE(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

DELETE	DELETE_E

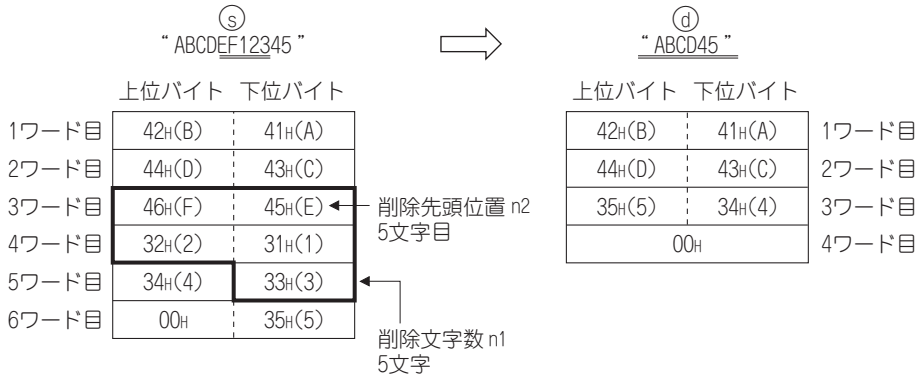
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_IN):	入力	:文字列(255)
	n1(_L):	削除文字数指定	:ワード［符号付き］
	n2(_P):	削除先頭位置指定	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:文字列(255)

機　能

演算処理

(1) ⓢへ入力された文字列の任意位置から指定文字数分のデータを削除して，残りの文字列をⓓから出力します。
削除する文字数は，n1 への入力値により指定します。
削除する文字列の先頭位置は，n2 への入力値により指定します。

（例）n1，n2 への入力値が，5 の場合



(2) ⓢへの入力値は，文字列型のデータ値で 0 ～ 255 バイトの範囲内です。

(3) n1 への入力値は，ワード [符号付き] 型のデータ値で 0 ～ 255 の範囲内です。
（ただし，ⓢへ入力される文字列の文字数以内に限ります。）

(4) n2 への入力値は，ワード [符号付き] 型のデータ値で 1 ～ 255 の範囲内です。
（ただし，ⓢへ入力される文字列の文字数以内に限ります。）

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。
その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

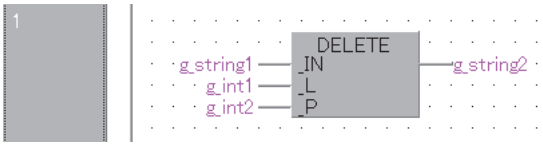
DELETE(_E) での演算エラーはありません。

プログラム例

ⓢへ入力された文字列の任意位置から指定文字数分のデータを削除して，残りの文字列をⓓから出力するプログラム。

(a) EN/ENO なし関数 (DELETE)

〔構造化ラダー/FBD〕

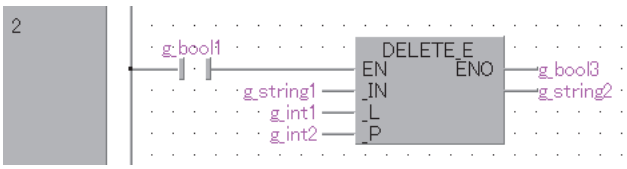


〔ST〕

```
g_string2:=DELETE(g_string1, g_int1, g_int2);
```

(b) EN/ENO 付き関数 (DELETE_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := DELETE_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

5.7.5 文字列の置換

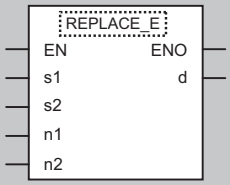
REPLACE(_E)

Basic High performance Process Redundant Universal LCPU

REPLACE(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= REPLACE_E (EN, s1, s2, n1, n2, d);
```

[ ] には，下記の関数が入ります。

REPLACE　　REPLACE_E

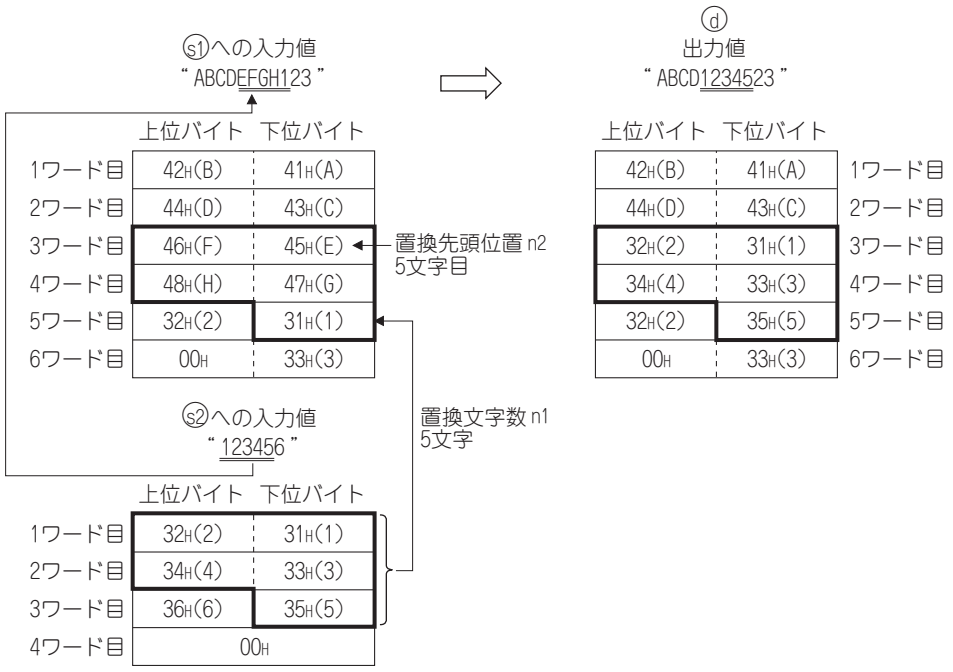
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	: ビット
	s1(_IN1): s2(_IN2):	入力	: 文字列 (255)
	n1(_L):	置換文字数指定	: ワード［符号付き］
	n2(_P):	置換先頭位置指定	: ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	: ビット
	d:	出力	: 文字列 (255)

機　能

演算処理

(1) ⓢ1 へ入力された文字列の任意位置から指定文字数分のデータを，ⓢ2 へ入力された文字列へ置換して，ⓓ から出力します。
置換する文字数は，n1 への入力値により指定します。
置換する文字列の先頭位置は，n2 への入力値により指定します。

（例）n1，n2 への入力値が，5 の場合



(2) ⓢ1，ⓢ2 への入力値は，文字列型のデータ値で 0 ～ 255 バイトの範囲内です。

(3) n1 への入力値は，ワード [符号付き] 型のデータ値で 0 ～ 255 の範囲内です。
（ただし，ⓢ1 へ入力される文字列の文字数以内に限ります。）

(4) n2 の入力値は，ワード [符号付き] 型のデータ値で 1 ～ 255 の範囲内です。
（ただし，ⓢ1 へ入力される文字列の文字数以内に限ります。）

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓ からは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓ から出力されるデータは不定値となります。
その場合は，ⓓ から出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

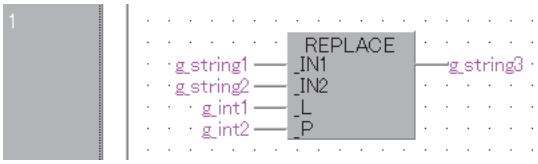
REPLACE(_E) での演算エラーはありません。

プログラム例

ⓢ1へ入力された文字列の任意位置から指定文字数分のデータを，ⓢ2へ入力された文字列へ置換して，ⓓから出力するプログラム

(a) EN/ENO なし関数 (REPLACE)

〔構造化ラダー/FBD〕

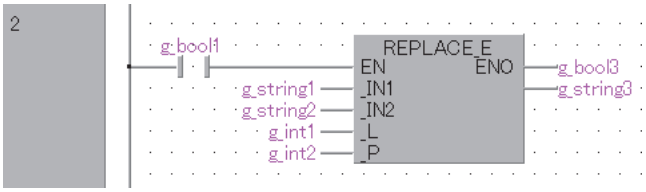


〔ST〕

```
g_string3:=REPLACE(g_string1, g_string2, g_int1, g_int2);
```

(b) EN/ENO 付き関数 (REPLACE_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := REPLACE_E(g_bool1, g_string1, g_string2, g_int1, g_int2,
g_string3);
```

5.8 時刻データ型関数

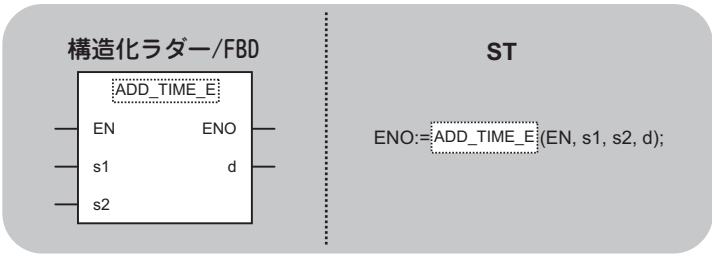
5.8.1 加算

ADD_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

ADD_TIME(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

ADD_TIME	ADD_TIME_E

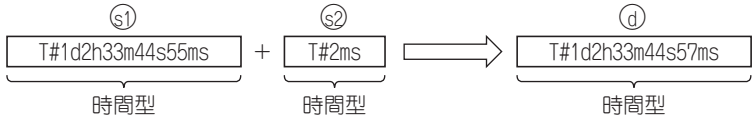
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	:時間
	s2(_IN2):		
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:時間

機　能

演算処理

(1) ⓢ1, ⓢ2 に入力された時間型データの加算（ⓢ1＋ⓢ2）を行い，演算結果を ⓓ から時間型で出力します。

（例）ⓢ1, ⓢ2 への入力値が，T#1d2h33m44s55ms(1 日 2 時間 33 分 44 秒 55 ミリ秒) と T#2ms(2 ミリ秒) の場合

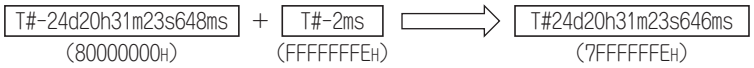


(2) ⓢ1, ⓢ2 への入力値は，時間型のデータ値です。

(3) 演算結果にアンダーフロー／オーバーフローが生じても演算エラーにはなりません。

ⓓへ下記のように出力されます。
また，ADD_TIME_E の場合，ENO から TRUE が出力されます。
（例）オーバーフロー

最上位ビットが 1 になるため，負の時間となります。
（例）アンダーフロー



最上位ビットが 0 になるため，正の時間となります。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

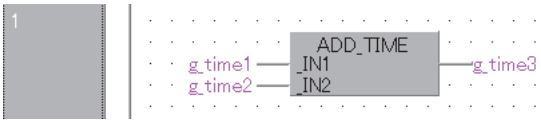
ADD_TIME(_E) での演算エラーはありません。

プログラム例

ⓢ1, ⓢ2に入力された時間型データの加算（ⓢ1＋ⓢ2）を行い，演算結果をⓓから時間型で出力するプログラム。

(a) EN/ENO なし関数 (ADD_TIME)

〔構造化ラダー/FBD〕

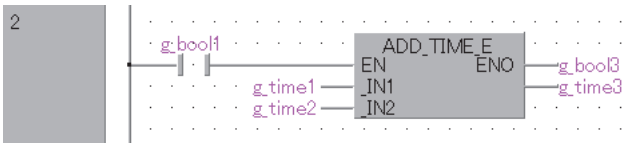


〔ST〕

```
g_time3:= ADD_TIME(g_time1, g_time2);
```

(b) EN/ENO 付き関数 (ADD_TIME_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := ADD_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

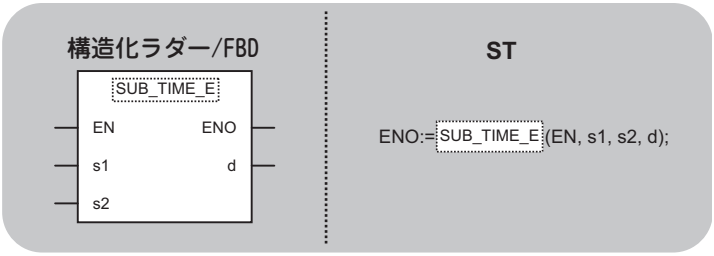
5.8.2 減算

SUB_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

SUB_TIME(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

SUB_TIME	SUB_TIME_E

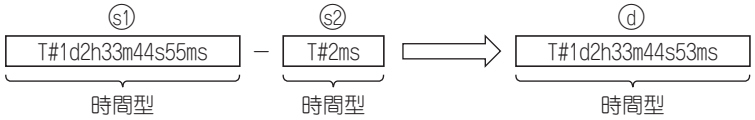
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	:時間
	s2(_IN2):		
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:時間

機　能

演算処理

(1) ⓢ1, ⓢ2 に入力された時間型データの減算（ⓢ1−ⓢ2）を行い，演算結果を ⓓ から時間型で出力します。

（例）ⓢ1, ⓢ2 への入力値が，T#1d2h33m44s55ms(1 日 2 時間 33 分 44 秒 55 ミリ秒) と T#2ms(2 ミリ秒) の場合



(2) ⓢ1, ⓢ2 への入力値は，時間型のデータ値です。

(3) 演算結果にオーバーフロー／アンダーフローが生じても演算エラーにはなりません。

ⓓへ下記のように出力されます。
また，SUB_TIME_E の場合，ENO から TRUE が出力されます。
（例）オーバーフロー

最上位ビットが 1 になるため，負の時間となります。

（例）アンダーフロー

最上位ビットが 0 になるため，正の時間となります。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

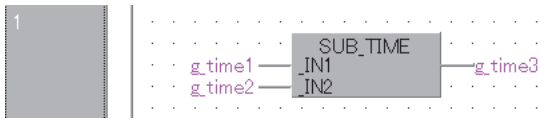
SUB_TIME(_E) での演算エラーはありません。

プログラム例

ⓢ1, ⓢ2に入力された時間型データの減算（ⓢ1−ⓢ2）を行い，演算結果をⓓから時間型で出力するプログラム。

(a) EN/ENO なし関数 (SUB_TIME)

〔構造化ラダー/FBD〕

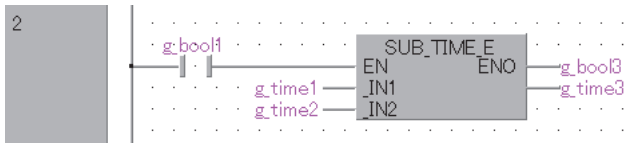


〔ST〕

```
g_time3:= SUB_TIME(g_time1, g_time2);
```

(b) EN/ENO 付き関数 (SUB_TIME_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := SUB_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

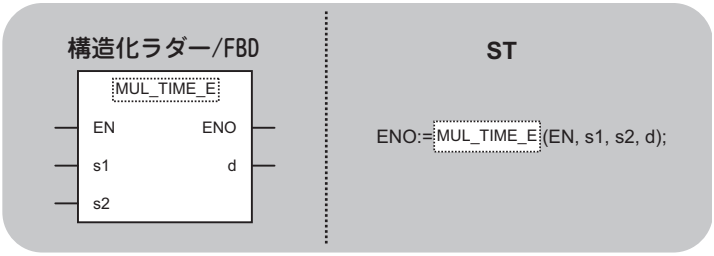
5.8.3 乗算

MUL_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

MUL_TIME(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

MUL_TIME	MUL_TIME_E

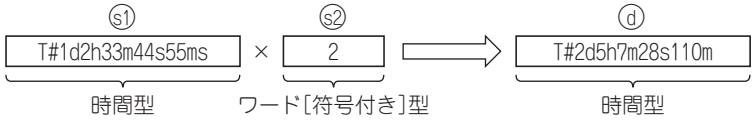
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	:時間
	s2(_IN2):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:時間

機　能

演算処理

(1) ⓢ1に入力された時間型データと，ⓢ2に入力されたワード［符号付き］型／ダブルワード［符号付き］型／単精度実数型／倍精度実数型データの乗算（ⓢ1×ⓢ2）を行い，演算結果をⓓから時間型で出力します。

（例）ⓢ1，ⓢ2への入力値が，T#1d2h33m44s55ms(1 日 2 時間 33 分 44 秒 55 ミリ秒) と 2 の場合



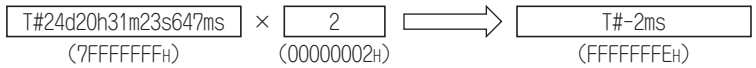
(2) ⓢ1への入力値は，時間型のデータ値です。

(3) ⓢ2への入力値は，ワード［符号付き］型 / ダブルワード［符号付き］型 / 単精度実数型 / 倍精度実数型のデータです。

(4) プログラミングツールからⓢ2に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

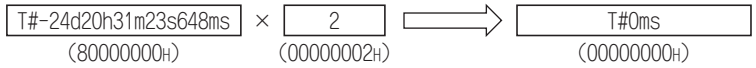
(5) 演算結果にオーバーフロー／アンダーフローが生じても演算エラーにはなりません。

ⓓへ下記のように出力されます。
また，MUL_TIME_E の場合，ENO から TRUE が出力されます。
（演算結果は 64 ビットデータになりますが，出力は上位 32 ビットを削除した時間型のデータで出力されます。）
（例）オーバーフロー



最上位ビットが 1 になるため，負の時間となります。

（例）アンダーフロー



最上位ビットが 0 になるため，正の時間となります。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

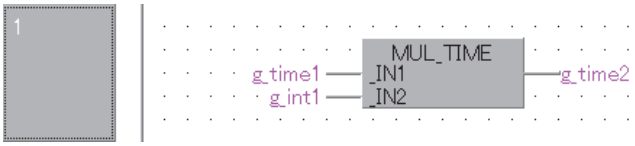
MUL_TIME(_E) での演算エラーはありません。

プログラム例

ⓢ1に入力された時間型データとⓢ2に入力されたワード［符号付き］型データの乗算（ⓢ1×ⓢ2）を行い，演算結果をⓓから時間型で出力するプログラム。

(a) EN/ENO なし関数 (MUL_TIME)

〔構造化ラダー/FBD〕

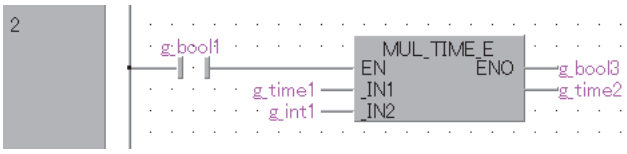


〔ST〕

```
g_time2:= MUL_TIME(g_time1, g_int1);
```

(b) EN/ENO 付き関数 (MUL_TIME_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := MUL_TIME_E(g_bool1, g_time1, g_int1, g_time2);
```

5.8.4 除算

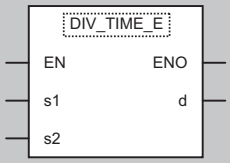
DIV_TIME(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

DIV_TIME(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= DIV_TIME_E (EN, s1, s2, d);
```

[ ] には，下記の関数が入ります。

DIV_TIME	DIV_TIME_E

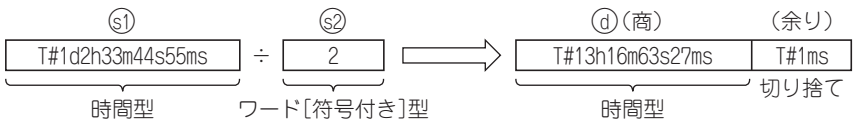
入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_IN1):	入力	:時間
	s2(_IN2):	入力	: ANY_NUM
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常）	:ビット
	d:	出力	:時間

機　能

演算処理

(1) ⓢ1に入力された時間型データと，ⓢ2に入力されたワード［符号付き］型／ダブルワード［符号付き］型／単精度実数型／倍精度実数型データの除算（ⓢ1 ÷ ⓢ2）を行い，演算結果の商をⓓから時間型で出力します。余りは切り捨てます。

（例）ⓢ1，ⓢ2への入力値が，T#1d2h33m44s55ms(1 日 2 時間 33 分 44 秒 55 ミリ秒) と 2 の場合



(2) ⓢ1への入力値は，時間型のデータ値です。

(3) ⓢ2への入力値は，ワード［符号付き］型 / ダブルワード［符号付き］型 / 単精度実数型 / 倍精度実数型のデータです。

（ただし，ⓢ2へ入力される値は，0 以外です。）

(4) プログラミングツールから㉂に単精度実数型／倍精度実数型データを指定した場合，丸め誤差が発生する場合があります。
プログラミングツールから入力値を設定する場合の注意事項は，MELSEC-Q/L/F 構造化プログラミングマニュアル（基礎編）を参照してください。

演算結果

(1) EN/ENO なし関数
演算処理を実行します。ⓓからは，演算値が出力されます。

(2) EN/ENO 付き関数
実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE *1	不定値

*1: ENO から FALSE が出力された場合，ⓓから出力されるデータは不定値となります。その場合は，ⓓから出力されたデータを使用しないようなプログラムの処理を行ってください。

エラー

次の場合には演算エラーとなります。

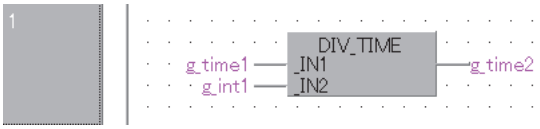
- ㉂へ入力される値が 0 のとき。（0 割り）　（エラーコード 4100）

プログラム例

㉀に入力された時間型データと㉂に入力されたワード [符号付き] 型データの除算（㉀÷㉂）を行い，演算結果の商をⓓから時間型で出力するプログラム。

(a) EN/ENO なし関数 (DIV_TIME)

〔構造化ラダー/FBD〕

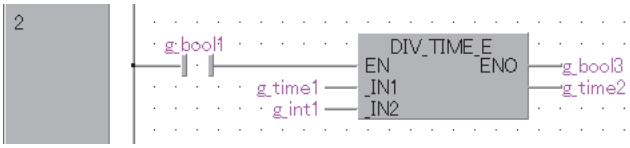


〔ST〕

```
g_time2:= DIV_TIME(g_time1, g_int1);
```

(b) EN/ENO 付き関数 (DIV_TIME_E)

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool3 := DIV_TIME_E(g_bool1, g_time1, g_int1, g_time2);
```

5.9 二安定ファンクションブロック

5.9.1 二安定ファンクションブロック（セット優先）

SR(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

SR(_E)

（_E : EN/ENO 付き）

構造化ラダー/FBD

ST

```
SR_E (EN, s1, s2, ENO, d);
```

[ ] には，下記の関数が入ります。

SR	SR_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_S1): s2(RESET):	入力	:ビット
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d(Q1):	出力	:ビット

☆機　能

演算処理

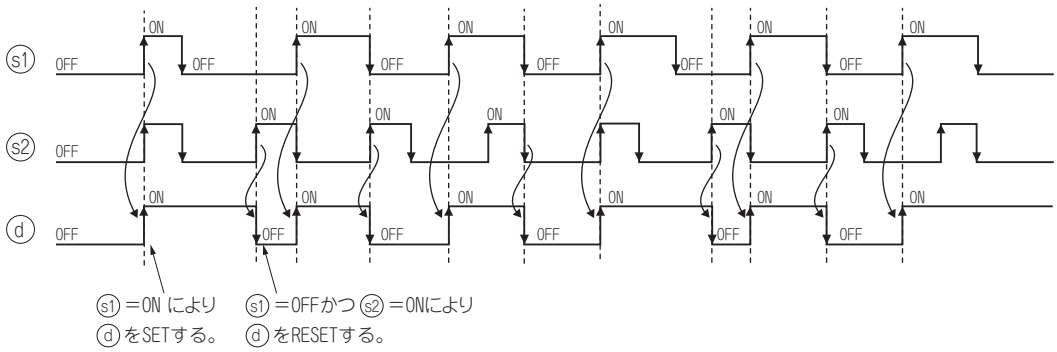
ⓢ1が ON すると，ⓓを SET します。ⓢ1が OFF の時，ⓢ2を ON すると，ⓓを RESET できます。
ⓢ1が ON のとき，ⓢ2を ON しても，ⓓは RESET されません。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ からは，演算値が出力されます。

〔タイミングチャート〕

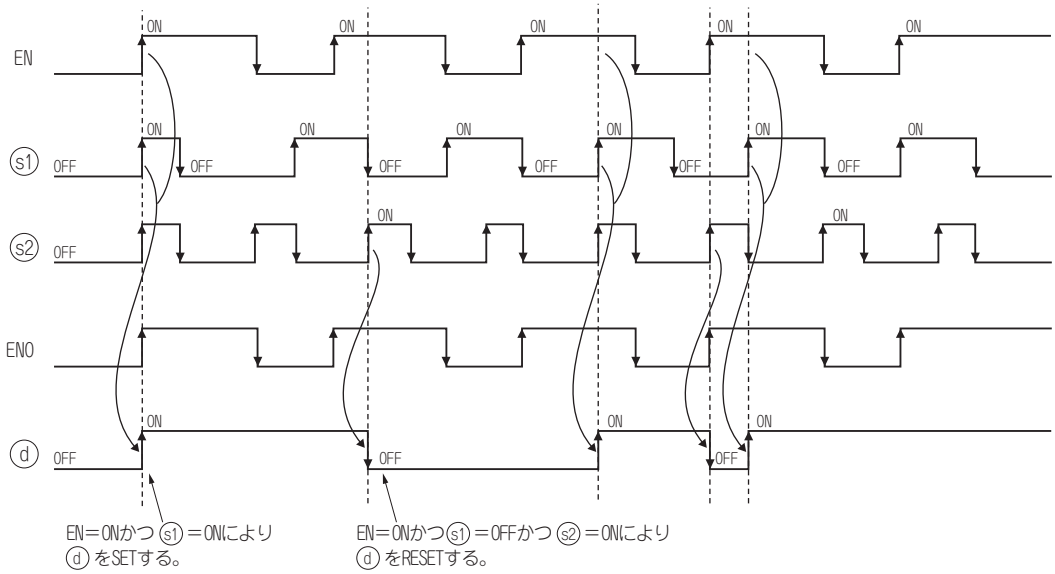


(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

〔タイミングチャート〕



エラー

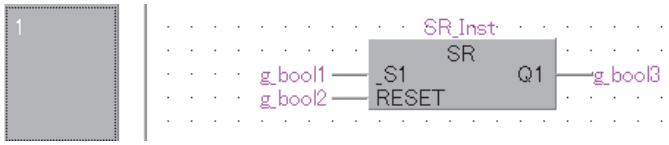
SR (_E) での演算エラーはありません。

プログラム例

ⓢ1に入力されたビット型のデータをⓓから出力保持し，ⓢ2に入力されたビット型のデータが 1 で，かつⓢ1の入力値が 0 の場合のみ，ⓓの値をリセットするプログラム。

(a) EN/ENO なし関数 (SR)

〔構造化ラダー/FBD〕

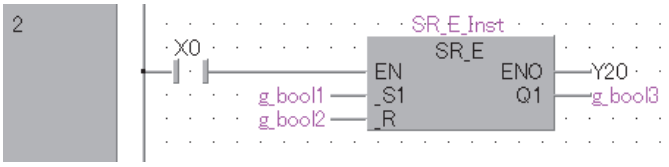


〔ST〕

```
SR_Inst(_S1:= g_bool1 ,RESET:= g_bool2 ,Q1:= g_bool3 );
```

(b) EN/ENO 付き関数 (SR_E)

〔構造化ラダー/FBD〕



〔ST〕

```
SR_E_Inst(EN:= X0 ,_S1:= g_bool1 ,_R:= g_bool2 ,Q1:= g_bool3 ,ENO:= Y20 );
```

5.9.2 二安定ファンクションブロック（リセット優先）

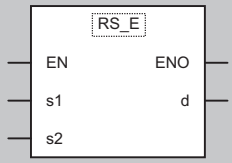
RS(_E)

Basic High performance Process Redundant Universal LCPU

RS(_E)

（ _E : EN/ENO 付き ）

構造化ラダー/FBD



ST

RS_E (EN, s1, s2, ENO, d);

[ ] には，下記の関数が入ります。

RS　　RS_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(_S): s2(_R1):	入力	:ビット
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d(Q1):	出力	:ビット

機　能

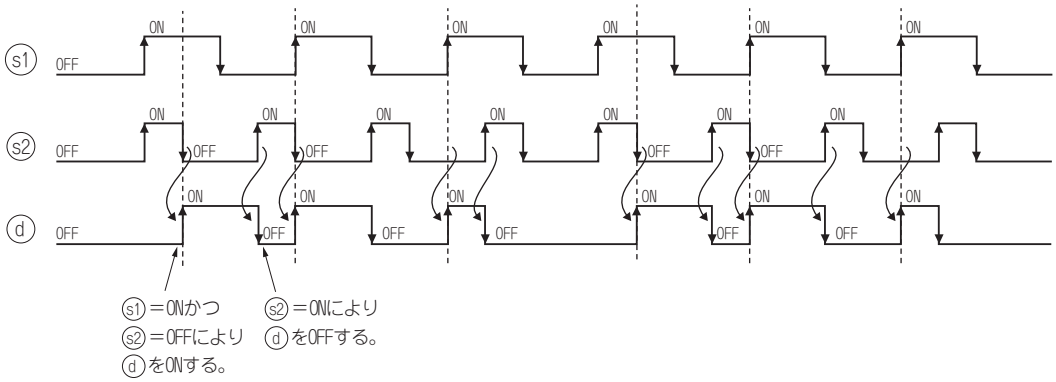
演算処理

ⓢ1 が ON すると，ⓓを SET します。ⓢ2 を ON すると，ⓓを RESET します。
ⓢ2 が ON のとき，ⓢ1 を ON しても，ⓓは SET できません。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。
〔タイミングチャート〕

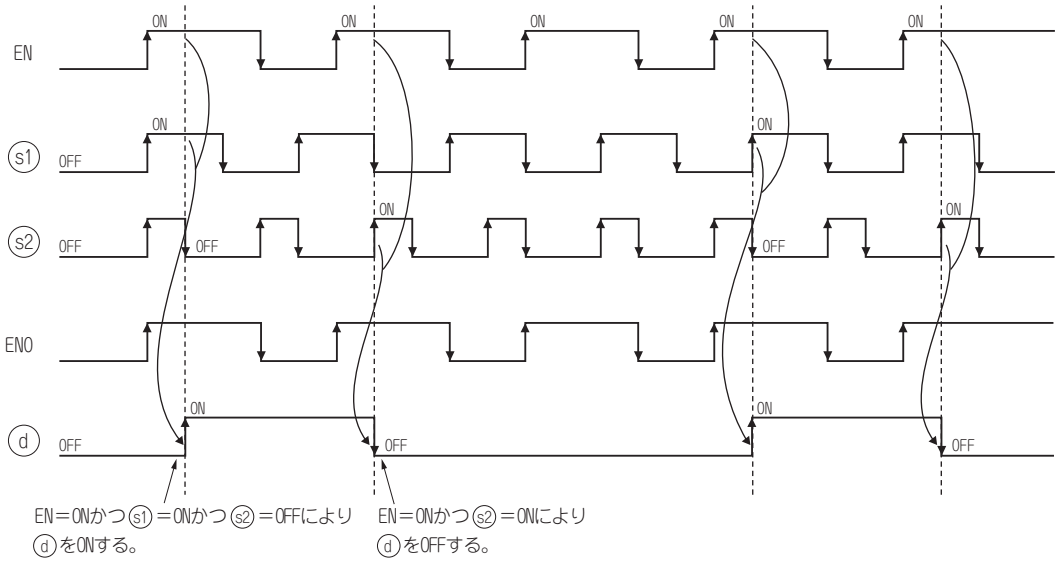


(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

〔タイミングチャート〕



エラー

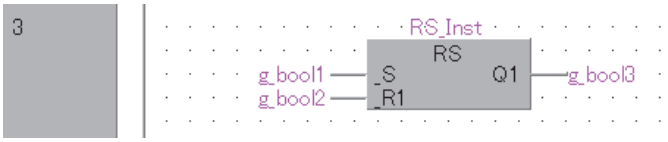
RS (_E) での演算エラーはありません。

プログラム例

ⓢ1に入力されたビット型のデータをⓓから出力保持し，ⓢ2に入力されたビット型のデータが 1 の時に強制的に，ⓓの値をリセットするプログラム。

(a) EN/ENO なし関数 (RS)

〔構造化ラダー/FBD〕

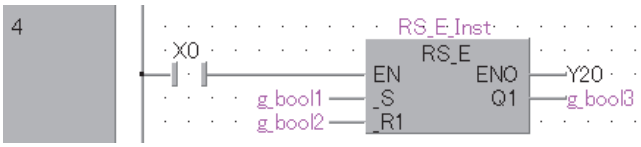


〔ST〕

```
RS_Inst(_S:= g_bool1 ,_R1:= g_bool2 ,Q1:= g_bool3 );
```

(b) EN/ENO 付き関数 (RS_E)

〔構造化ラダー/FBD〕



〔ST〕

```
RS_E_Inst(EN:= X0 ,_S:= g_bool1 ,_R1:= g_bool2 ,Q1:= g_bool3 ,ENO:= Y20 );
```

5.10 エッジ検出ファンクションブロック

5.10.1 立ち上りエッジ検出

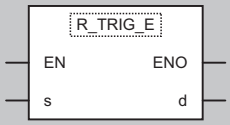
R_TRIG(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

R_TRIG(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
R_TRIG_E (EN, s, ENO, d);
```

[ ] には，下記の関数が入ります。

R_TRIG	R_TRIG_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_CLK):	入力	:ビット
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d(Q):	出力	:ビット

機　能

演算処理

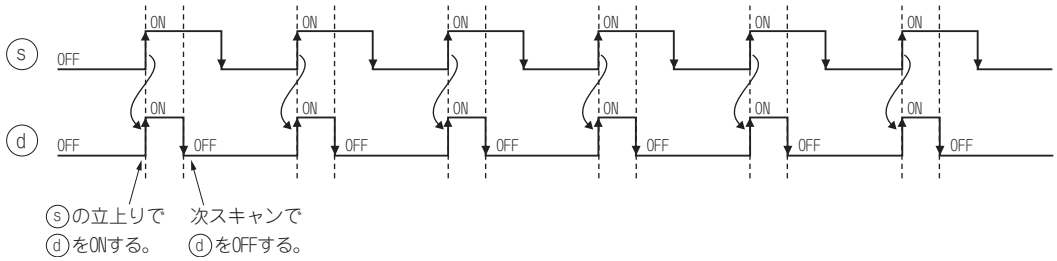
ⓢが ON すると，ⓓを 1 スキャンのみ ON します。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

〔タイミングチャート〕

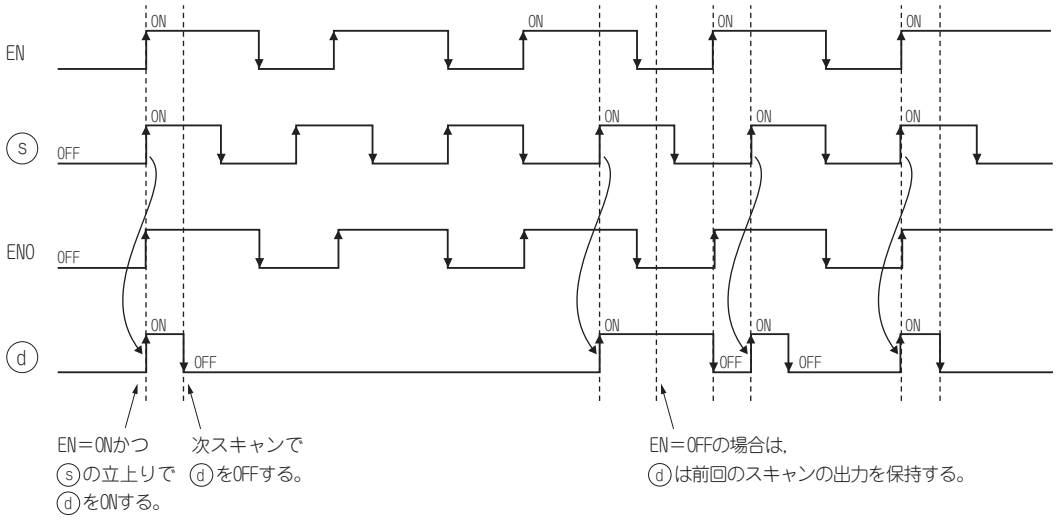


(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

〔タイミングチャート〕



エラー

R_TRIG (_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータが OFF→ON に変化したとき, ⓓを 1 スキャンのみ ON する。

(a) EN/ENO なし関数 (R_TRIG)

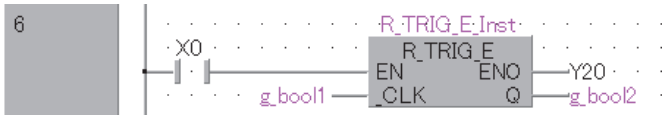
〔構造化ラダー/FBD〕

〔ST〕

```
R_TRIG_Inst(_CLK:= g_bool1 ,Q:= g_bool2 );
```

(b) EN/ENO 付き関数 (R_TRIG_E)

〔構造化ラダー/FBD〕



〔ST〕

```
R_TRIG_E_Inst(EN:= X0 ,_CLK:= g_bool1 ,Q:= g_bool2 ,ENO:= Y20 );
```

5.10.2 立ち下りエッジ検出

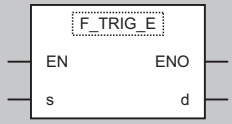
F_TRIG(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

F_TRIG(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
F_TRIG_E (EN, s, ENO, d);
```

[ ] には，下記の関数が入ります。

F_TRIG	F_TRIG_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(_CLK):	入力	:ビット
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d(Q):	出力	:ビット

機　能

演算処理

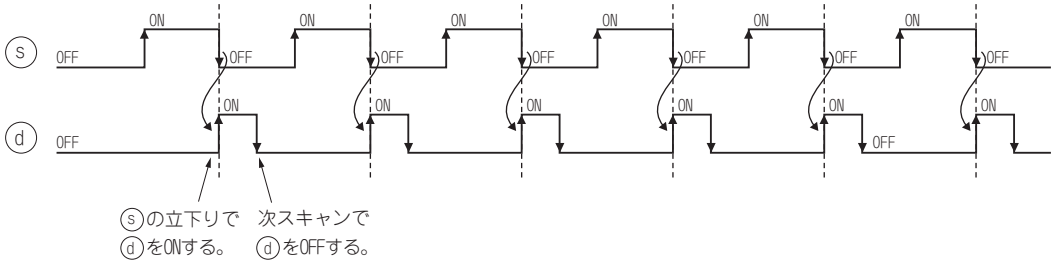
ⓢが OFF すると，ⓓを 1 スキャンのみ ON します。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓからは，演算値が出力されます。

〔タイミングチャート〕

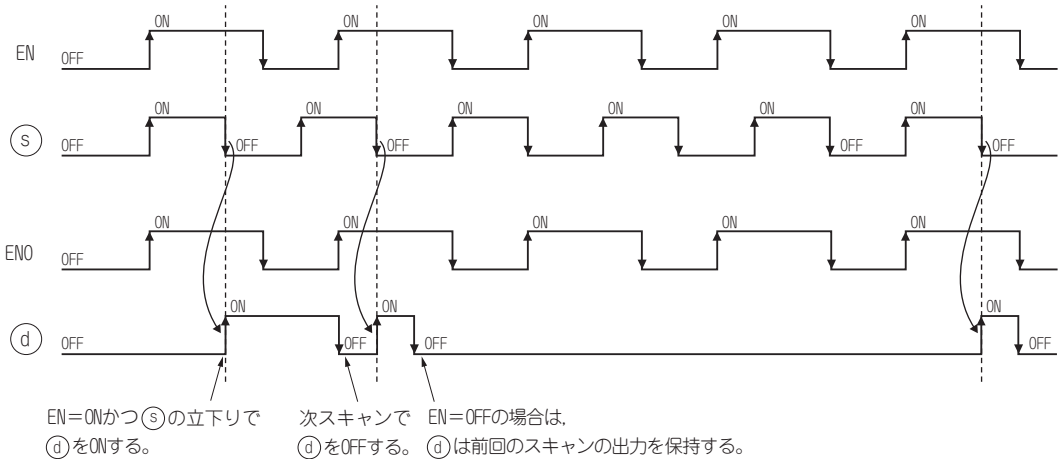


(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

〔タイミングチャート〕



エラー

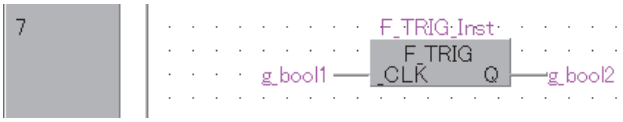
F_TRIG (_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータが ON→OFF に変化したとき，ⓓを 1 スキャンのみ ON する。

(a) EN/ENO なし関数 (F_TRIG)

〔構造化ラダー/FBD〕

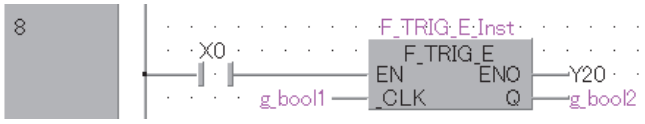


〔ST〕

```
F_TRIG_Inst(_CLK:= g_bool1 ,Q:= g_bool2 );
```

(b) EN/ENO 付き関数 (F_TRIG_E)

〔構造化ラダー/FBD〕



〔ST〕

```
F_TRIG_E_Inst(EN:= X0 ,_CLK:= g_bool1 ,Q:= g_bool2 ,ENO:= Y20 );
```

5.11 カウンタファンクションブロック

5.11.1 アップカウンタ

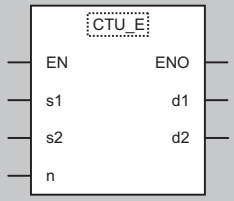
CTU(_E)

Basic High performance Process Redundant Universal LCPU

CTU(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
CTU_E (EN, s1, s2, n, ENO, d1, d2);
```

[ ] には，下記の関数が入ります。

CTU	CTU_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(CU):	カウント信号入力	:ビット
	s2(RESET):	カウントリセット	:ビット
	n(PV):	カウント最大値	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d1(Q):	カウント一致出力	:ビット
	d2(CV):	カウント値	:ワード［符号付き］

機　能

演算処理

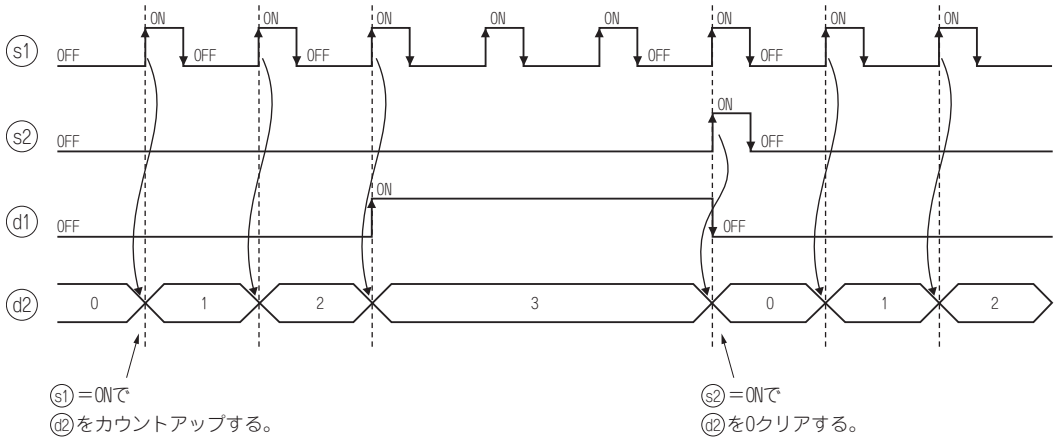
(1) ⓢ1が ON すると，ⓓ2をカウントします。
カウント値ⓓ2が n に達するとⓓ1が ON します。
ⓢ2を ON するとⓓ1を OFF し，カウント値ⓓ2をリセットします。

(2) n の有効設定範囲は，-32768 ～ 32767 です。
ただし，0 以下の数値を設定した場合，ⓓ2のカウント値に関わらずⓓ1は ON します。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ1, ⓓ2からは，演算値が出力されます。
[タイミングチャート]

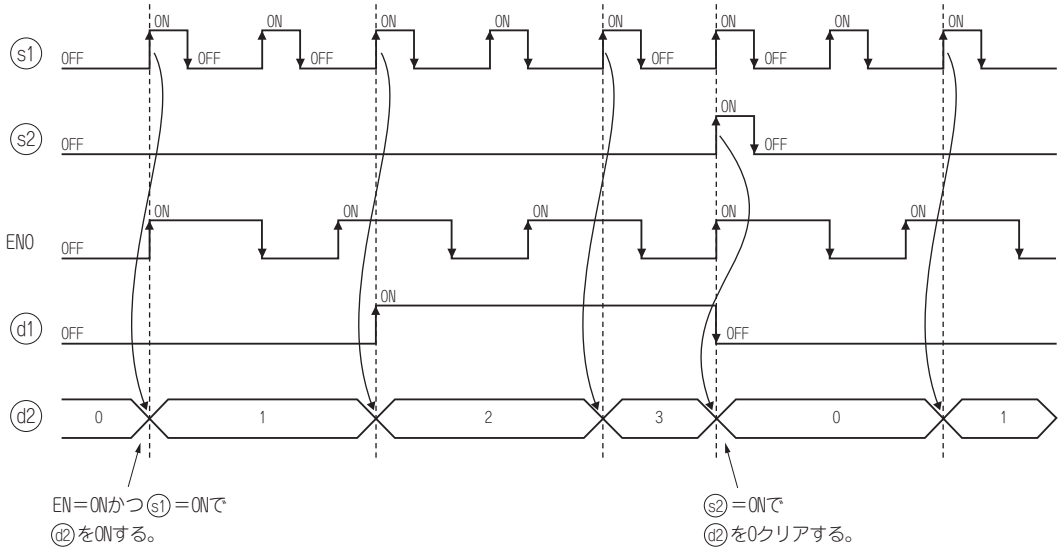


(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ1, ⓓ2
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

[タイミングチャート]



エラー

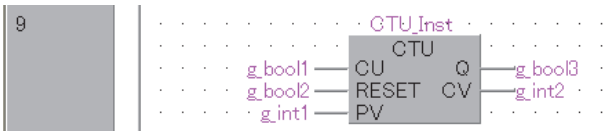
CTU (_E) での演算エラーはありません。

プログラム例

ⓢ1に入力されたビット型のデータが OFF→ON に変化した回数を，ⓓ2から出力するプログラム。

(a) EN/ENO なし関数 (CTU)

〔構造化ラダー/FBD〕

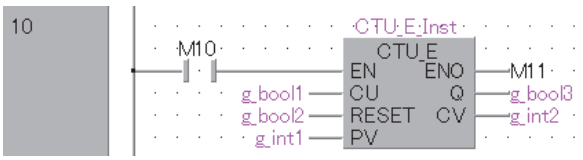


〔ST〕

```
CTU_Inst(CU:= g_bool1 ,RESET:= g_bool2 ,PV:= g_int1 ,Q:= g_bool3
,CV:= g_int2 );
```

(b) EN/ENO 付き関数 (CTU_E)

〔構造化ラダー/FBD〕



〔ST〕

```
CTU_E_Inst(EN:= M10 ,CU:= g_bool1 ,RESET:= g_bool2 ,PV:= g_int1
,Q:= g_bool3 ,CV:= g_int2 ,ENO:= M11 );
```

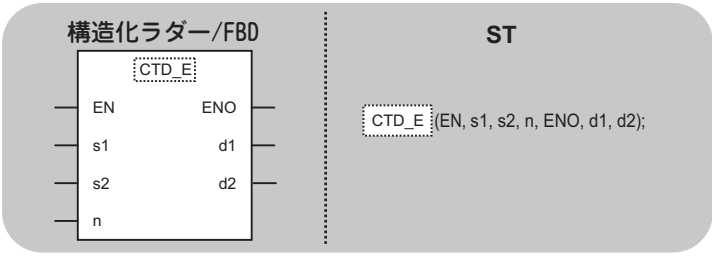
5.11.2 ダウンカウンタ

CTD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

CTD(_E)

(_E : EN/ENO 付き)



CTD_E には，下記の関数が入ります。

CTD	CTD_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(CD):	カウント信号入力	:ビット
	s2(LOAD):	カウントリセット	:ビット
	n(PV):	カウント開始値	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d1(Q):	カウント一致出力	:ビット
	d2(CV):	カウント値	:ワード［符号付き］

機　能

演算処理

(1) ⓢ1が OFF→ON すると，ⓓ2を減算カウント（–1）します。
n はカウントの減算初期値をセットします。
カウント値ⓓ2が 0 の場合，ⓓ1が ON します。
ⓢ2を ON するとⓓ1を OFF し，カウント値ⓓ2に減算初期値 n をセットします。

(2) n の有効設定範囲は，-32768 ～ 32767 です。
ただし，0 以下の数値を設定した場合，ⓓ2のカウント値に関わらずⓓ1は ON します。

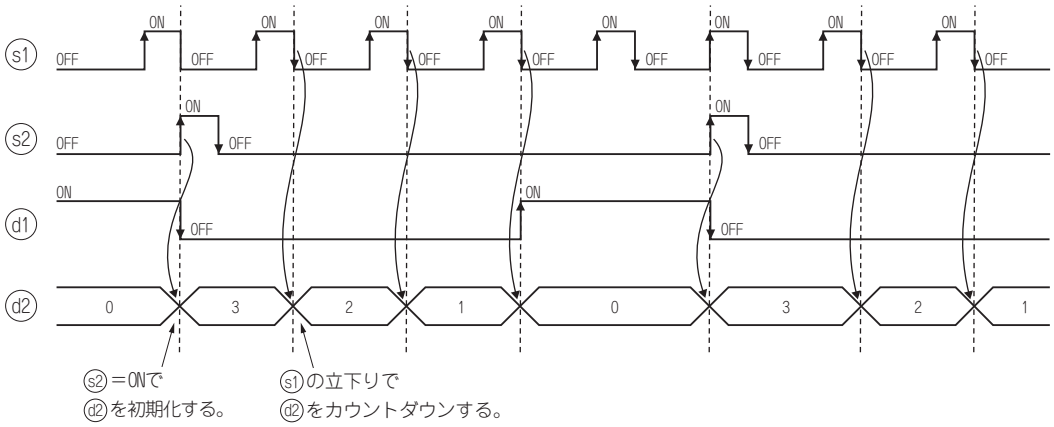
演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ1, ⓓ2からは，演算値が出力されます。

[タイミングチャート]

n=3の場合



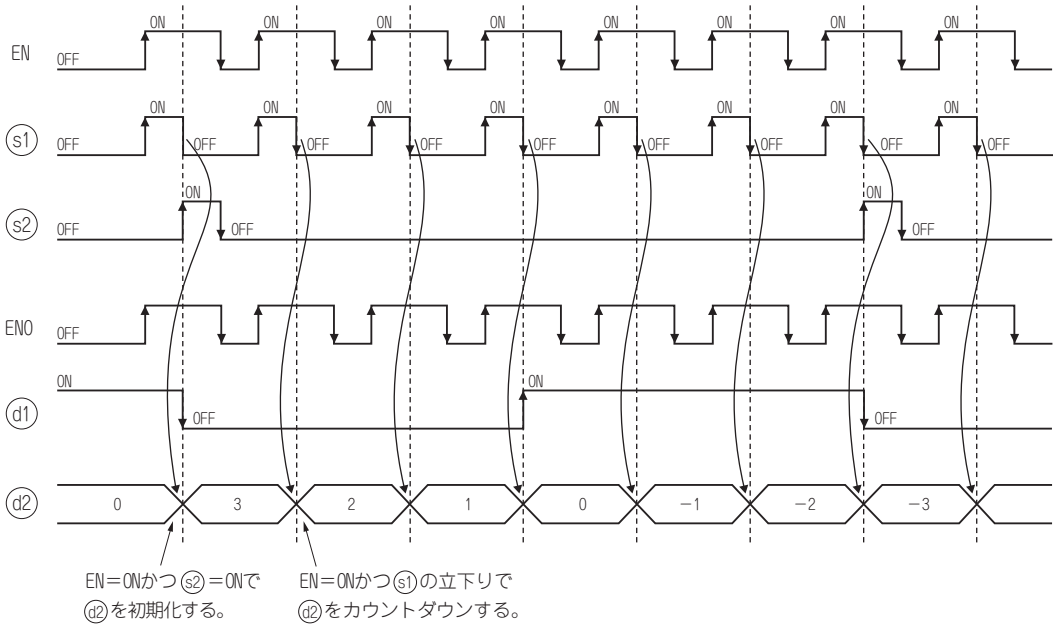
(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ1, ⓓ2
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

[タイミングチャート]

n=3の場合



エラー

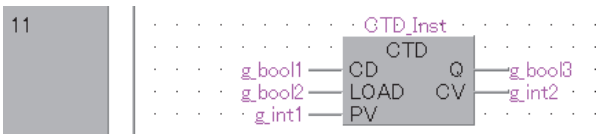
CTD (_E) での演算エラーはありません。

プログラム例

ⓢ1に入力されたビット型のデータが OFF→ON に変化した回数をカウントし, ⓓ2の値が 0 になったらⓓ1を ON するプログラム。

(a) EN/ENO なし関数 (CTD)

〔構造化ラダー/FBD〕

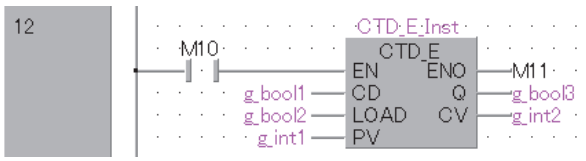


〔ST〕

```
CTD_Inst(CD:= g_bool1 ,LOAD:= g_bool2 ,PV:= g_int1 ,Q:= g_bool3
,CV:= g_int2 );
```

(b) EN/ENO 付き関数 (CTD_E)

〔構造化ラダー/FBD〕



〔ST〕

```
CTD_E_Inst(EN:= M10 ,CD:= g_bool1 ,LOAD:= g_bool2 ,PV:= g_int1
,Q:= g_bool3 ,CV:= g_int2 ,ENO:= M11 );
```

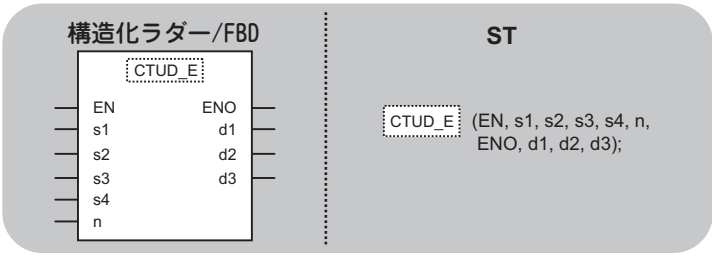
5.11.3 アップダウンカウンタ

CTUD(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

CTUD(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

CTUD	CTUD_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s1(CU):	カウントアップ信号入力	:ビット
	s2(CD):	カウントダウン信号入力	:ビット
	s3(RESET):	カウントアップリセット	:ビット
	s4(LOAD):	カウントダウンリセット	:ビット
	n(PV):	カウント最大値	:ワード［符号付き］
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d1(QU):	カウントアップ一致出力	:ビット
	d2(QD):	カウントダウン一致出力	:ビット
	d3(CV):	現在カウント値	:ワード［符号付き］

機　能

演算処理

(1) ⓢ1 が OFF→ON すると，ⓓ3 を加算カウント（+1）します。

ⓢ2 が OFF→ON すると，ⓓ3 を減算カウント（–1）します。

n はカウンタの最大値を設定します。

ⓓ3 が 0 の場合，ⓓ2 が ON します。

ⓓ3 がカウンタの最大値 n に達するとⓓ1 が ON します。

ⓢ3 を ON すると，ⓓ3 をリセットします。

ⓢ4 を ON すると n の値をⓓ3 にセットします。

(2) n の有効設定範囲は，-32768 ～ 32767 です。

ただし，0 以下の数値を設定した場合，ⓓ3 のカウント値に関わらずⓓ1，ⓓ2 は ON します。

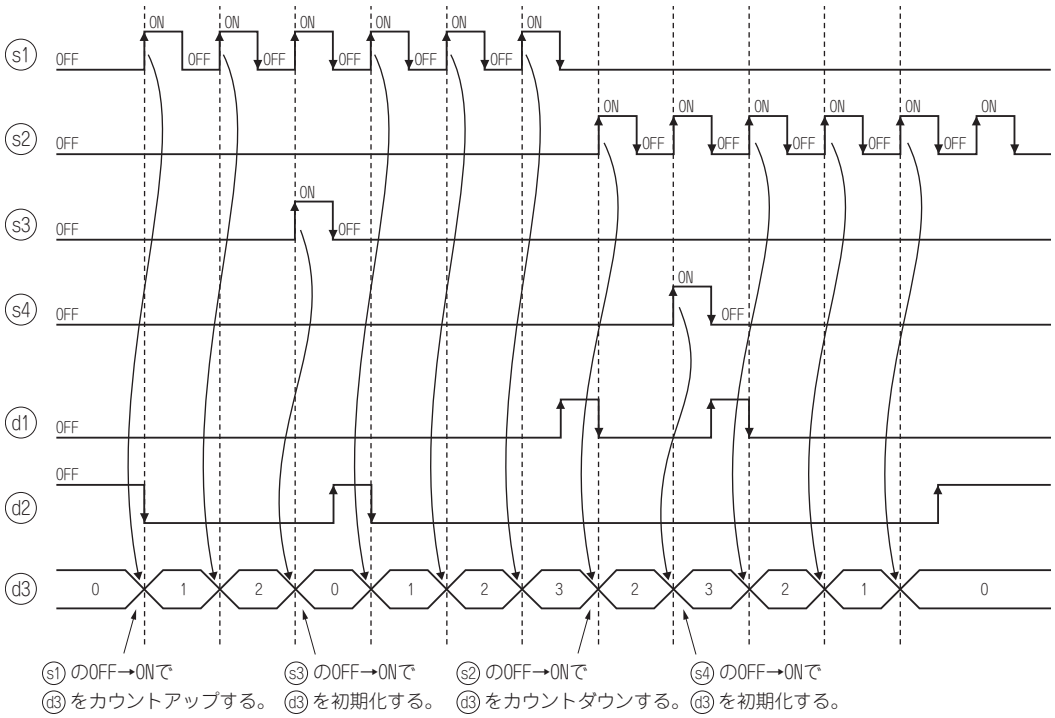
演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ1, ⓓ2, ⓓ3 からは，演算値が出力されます。

[タイミングチャート]

n=3の場合



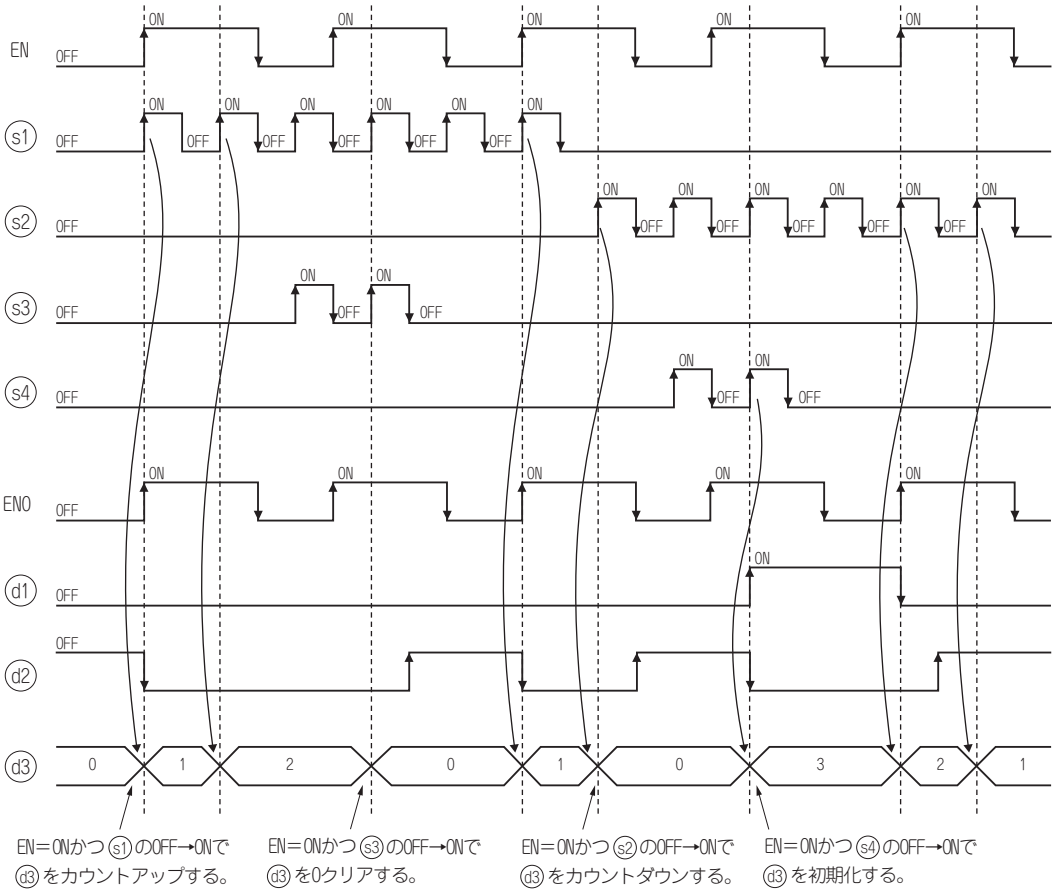
(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ1, ⓓ2, ⓓ3
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

[タイミングチャート]

n=3の場合



エラー

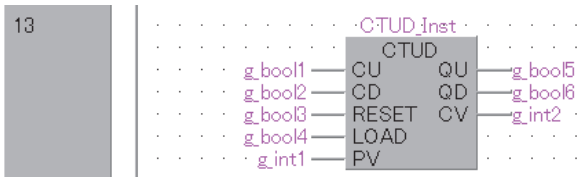
CTUD (_E) での演算エラーはありません。

プログラム例

ⓢ1に入力されたビット型のデータが OFF→ON に変化した回数をカウントし，ⓓ3の値がⓝと同じになったらⓓ1を ON するプログラム。また，同時にⓢ2に入力されたビット型のデータが OFF→ON に変化した回数をカウントし，ⓓ3の値が 0 になったらⓓ2を ON するプログラム。

(a) EN/ENO なし関数 (CTD)

〔構造化ラダー/FBD〕

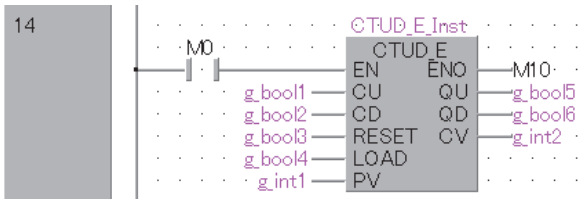


〔ST〕

```
CTUD_Inst(CU:= g_bool1 ,CD :=g_bool2 ,RESET:= g_bool3 ,LOAD:= g_bool4
,PV:= g_int1 ,QU:= g_bool5 ,QD:= g_bool6 ,CV:= g_int2 );
```

(b) EN/ENO 付き関数 (CTD_E)

〔構造化ラダー/FBD〕



〔ST〕

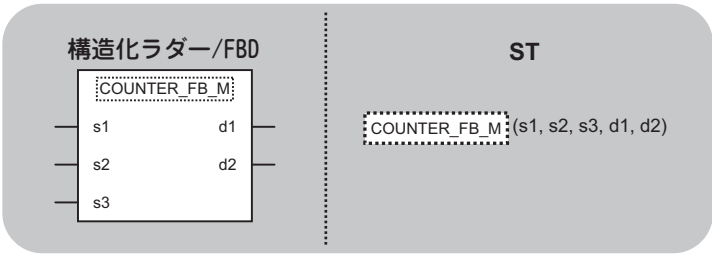
```
CTUD_E_Inst(EN:= M0 ,CU:= g_bool1 ,CD:= g_bool2 ,RESET:= g_bool3
,LOAD:= g_bool4 ,PV:= g_int1 ,QU:= g_bool5 ,QD:= g_bool6 ,CV:= g_int2
,ENO:= M10 );
```

5.11.4 カウンタファンクションブロック

COUNTER_FB_M

Basic | High performance | Process | Redundant | Universal | LCPU

COUNTER_FB_M



[ ] には，下記の関数が入ります。

COUNTER_FB_M

入力引数，	s1(Coil):	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s2(Preset):	カウンタ設定値	:ワード［符号付き］
	s3(ValueIn):	カウンタ初期値	:ワード［符号付き］
出力引数，	d1(ValueOut):	カウンタ現在値	: ANY16
	d2(Status):	出力	:ビット

機　能

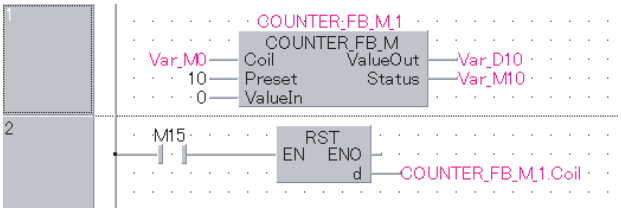
演算処理

(1) ⓢ1 の立上がり（OFF→ON）を検出してカウントします。ⓢ1 が ON のままではカウントされません。カウントは ⓢ3 の値から開始し，ⓢ2 の値になると，ⓓ2 が ON します。現在のカウント値は ⓓ1 に格納されます。

(2) ⓢ2 の有効設定範囲は，0 ～ 32767 の値が指定可能です。

(3) ⓢ3 の有効設定範囲は，-32768 ～ 32767 の値が指定可能です。ただし，負の値を指定した場合，0 が初期値となります。

(4) カウンタの現在値をリセットしたい場合は, ⓢ1 をリセットしてください。
(例) インスタンス名が COUNTER_FB_M_1 の場合
〔構造化ラダー/FBD〕



ⓢ1 (Coil) をリセットする

〔ST〕

```
COUNTER_FB_M_1(Coil:=Var_M0,Preset:=10,ValueIn:=0,ValueOut:=Var_D10,Sta
tus:=Var_M10);
RST(M15,COUNTER_FB_M_1.Coil);
```

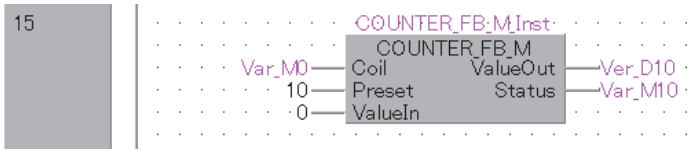
エラー

カウンタファンクションブロックでの演算エラーはありません。

プログラム例

ⓢ1 に入力されたビット型のデータが OFF→ON に変化した回数を, ⓓ1 から出力するプログラム。

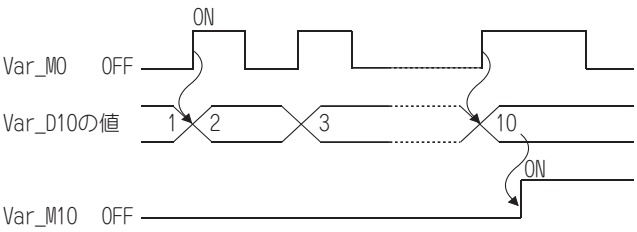
〔構造化ラダー/FBD〕



〔ST〕

```
    COUNTER_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 0
    ,ValueOut:= Var_D10 ,Status:= Var_M10 );
```

【タイミングチャート】



5.12 タイマファンクションブロック

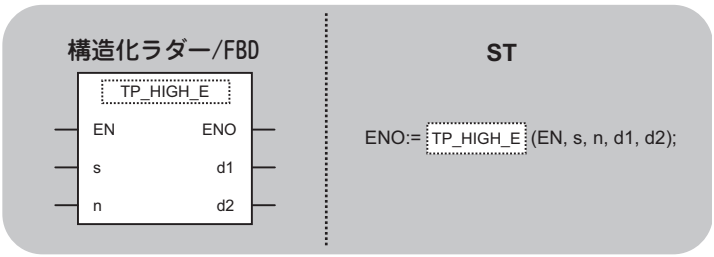
5.12.1 パルスタイマ

TP(_E), TP_HIGH(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

TP(_E)
TP_HIGH(_E)

(_E : EN/ENO 付き)



TP_HIGH_E には，下記の関数が入ります。

TP	TP_E
TP_HIGH	TP_HIGH_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(IN):	入力	:ビット
	n(PT):	出力時間設定値	:時間
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d1(Q):	出力	:ビット
	d2(ET):	経過時間	:時間

機　能

演算処理

ⓢが ON すると，n で設定された時間の間，ⓓ1を ON します。ⓓ2にはⓓ1が ON している経過時間がセットされます。

経過時間が設定時間に到達するとⓓ1は OFF します。

ⓓ1が OFF しても経過時間はリセットされません。

ⓓ1が OFF 後，ⓢが OFF の場合に経過時間をリセットします。

(1) TP(_E)

経過時間のカウントに低速タイマを使用します。
出力時間を 1ms ～ 1000ms の単位で設定できます。単位は PC パラメータの PC システム設定のタイマ時限設定で設定します。
n の有効設定範囲は，T#0ms ～ T#3276700ms です。

(2) TP_HIGH(_E)

経過時間のカウントに高速タイマを使用します。
出力時間を下記の単位で設定できます。単位は PC パラメータの PC システム設定のタイマ時限設定で設定します。

CPU ユニット	設定単位
ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU	0.1ms ～ 100ms
ユニバーサルモデル QCPU，LCPU	0.01ms ～ 100ms

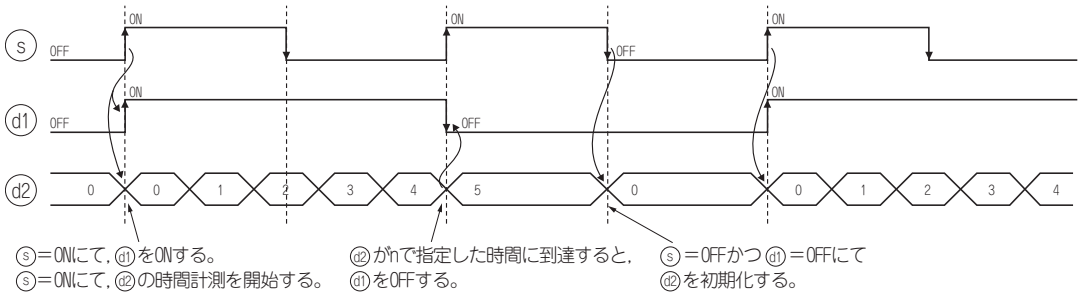
n の有効設定範囲は，T#0ms ～ T#327670ms です。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ1，ⓓ2 からは，演算値が出力されます。
[タイミングチャート]

n= T#5s（5秒）の場合



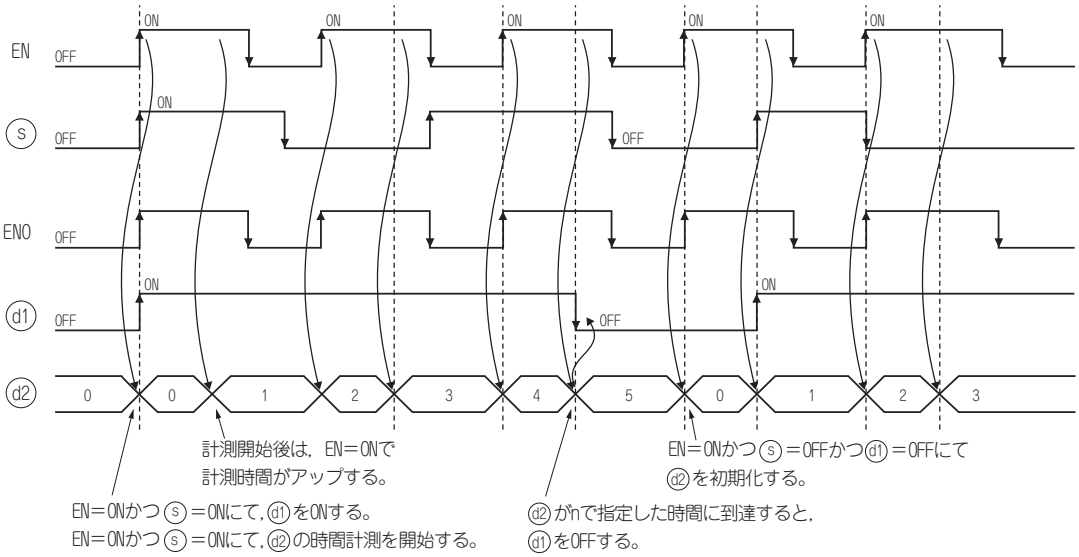
(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ1，ⓓ2
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

[タイミングチャート]

n＝T#5s（5秒）の場合



エラー

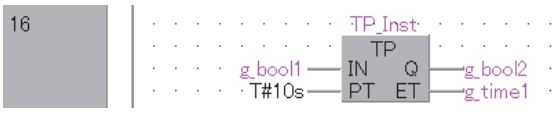
TP (_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータが ON したとき，ⓓ1のビット型のデータを 10 秒間 ON するプログラム。

(a) EN/ENO なし関数 (TP)

〔構造化ラダー/FBD〕

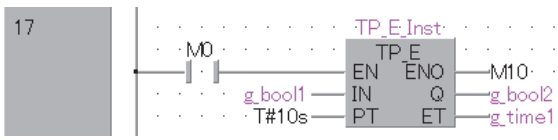


〔ST〕

```
TP_Inst(IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1 );
```

(b) EN/ENO 付き関数 (TP_E)

〔構造化ラダー/FBD〕



〔ST〕

```
TP_E_Inst(EN:= M0 ,IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1
,ENO:= M10 );
```

5.12.2 オンディレータイマ

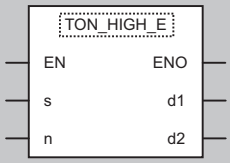
TON(_E), TON_HIGH(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

TON(_E)
TON_HIGH(_E)

(_E : EN/ENO 付き)

構造化ラダー/FBD



ST

```
ENO:= TON_HIGH_E (EN, s, n, d1, d2);
```

[ ] には，下記の関数が入ります。

TON	TON_E
TON_HIGH	TON_HIGH_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(IN):	入力	:ビット
	n(PT):	遅延時間設定値	:時間
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d1(Q):	出力	:ビット
	d2(ET):	経過時間	:時間

機　能

演算処理

ⓢが ON すると，n で設定された時間の後，ⓓ1を ON します。ⓓ2にはⓓ1が ON するまでの遅延経過時間がセットされます。

ⓢが OFF するとⓓ1を OFF し遅延経過時間もリセットします。

(1) TON(_E)

経過時間のカウントに低速タイマを使用します。
出力時間を 1ms ～ 1000ms の単位で設定できます。単位は PC パラメータの PC システム設定のタイマ時限設定で設定します。
n の有効設定範囲は，T#0ms ～ T#3276700ms です。

(2) TON_HIGH(_E)

経過時間のカウントに高速タイマを使用します。
出力時間を下記の単位で設定できます。単位は PC パラメータの PC システム設定のタイマ時限設定で設定します。

CPU ユニット	設定単位
ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU	0.1ms ～ 100ms
ユニバーサルモデル QCPU，LCPU	0.01ms ～ 100ms

n の有効設定範囲は，T#0ms ～ T#327670ms です。

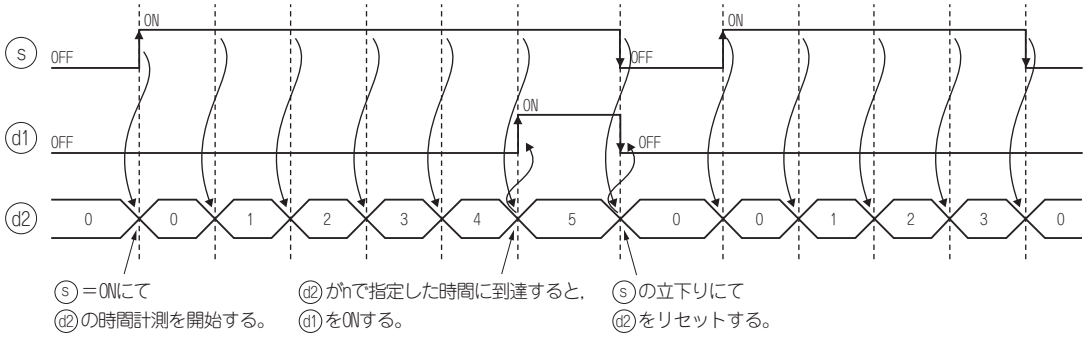
演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ1，ⓓ2 からは，演算値が出力されます。

[タイミングチャート]

n= T#5s（5秒）の場合



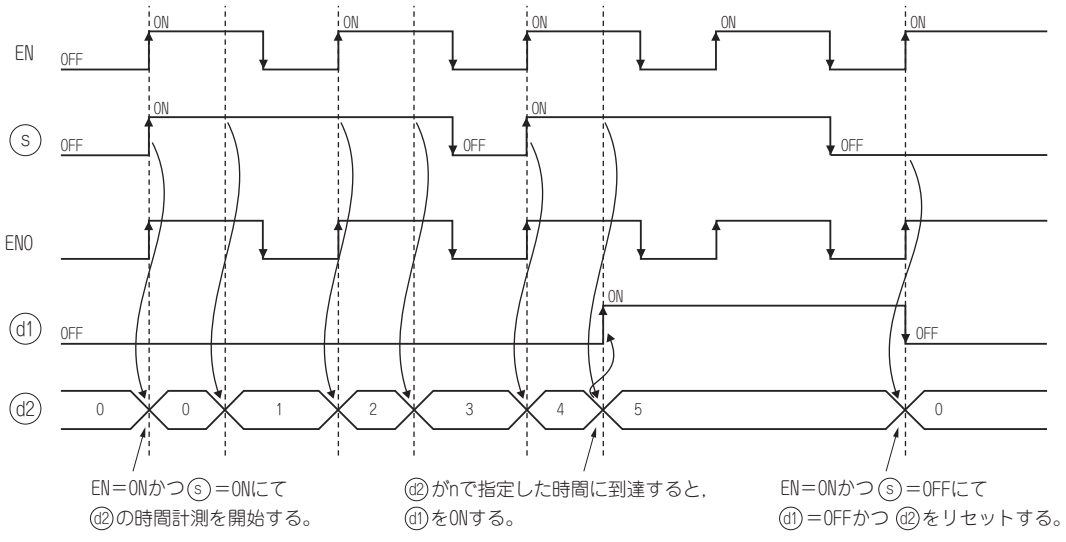
(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ1，ⓓ2
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

[タイミングチャート]

n= T#5s（5秒）の場合



エラー

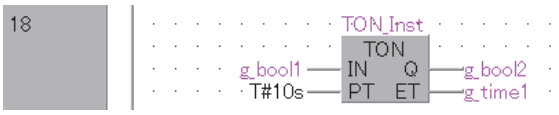
TON (_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータが ON したとき，10 秒後にⓓ1のビット型のデータを ON するプログラム。

(a) EN/ENO なし関数 (TON)

〔構造化ラダー/FBD〕

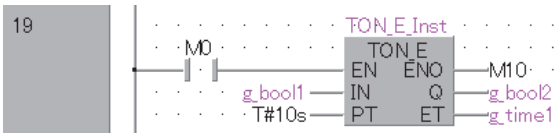


〔ST〕

```
TON_Inst(IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1 );
```

(b) EN/ENO 付き関数 (TON_E)

〔構造化ラダー/FBD〕



〔ST〕

```
TON_E_Inst(EN:= M0 ,IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1
,ENO:= M10 );
```

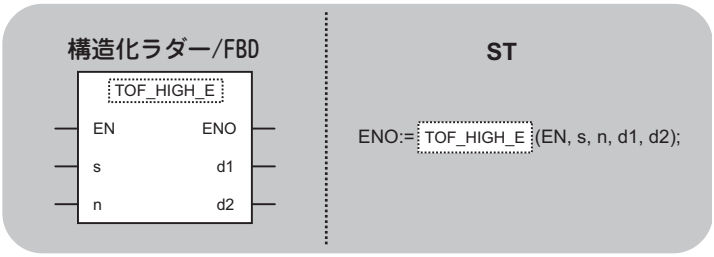
5.12.3 オフディレータイマ

TOF(_E), TOF_HIGH(_E)

Basic | High performance | Process | Redundant | Universal | LCPU

TOF(_E)
TOF_HIGH(_E)

(_E : EN/ENO 付き)



[ ] には，下記の関数が入ります。

TOF	TOF_E
TOF_HIGH	TOF_HIGH_E

入力引数，	EN:	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s(IN):	入力	:ビット
	n(PT):	遅延時間設定値	:時間
出力引数，	ENO:	実行結果（TRUE：正常　FALSE：異常または停止）	:ビット
	d1(Q):	出力	:ビット
	d2(ET):	経過時間	:時間

☆ 機　能

演算処理

ⓢが ON すると，ⓓ1を ON します。

ⓢが ON→OFF すると，n で設定された時間の後，ⓓ1を OFF します。ⓓ2にはⓓ1が OFF するまでの経過時間がセットされます。

ⓢが再び ON するとⓓ1を ON し，経過時間をリセットします。

(1) TOF(_E)

経過時間のカウントに低速タイマを使用します。
出力時間を 1ms ～ 1000ms の単位で設定できます。単位は PC パラメータの PC システム設定のタイマ時限設定で設定します。
n の有効設定範囲は，T#0ms ～ T#3276700ms です。

(2) TOF_HIGH(_E)

経過時間のカウントに高速タイマを使用します。
出力時間を下記の単位で設定できます。単位は PC パラメータの PC システム設定のタイマ時限設定で設定します。

CPU ユニット	設定単位
ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU	0.1ms ～ 100ms
ユニバーサルモデル QCPU，LCPU	0.01ms ～ 100ms

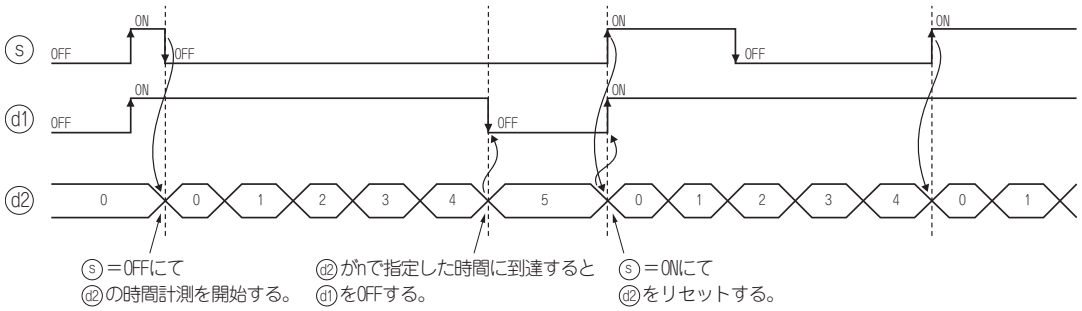
n の有効設定範囲は，T#0ms ～ T#327670ms です。

演算結果

(1) EN/ENO なし関数

演算処理を実行します。ⓓ1，ⓓ2 からは，演算値が出力されます。
[タイミングチャート]

n＝T#5s(5秒)の場合



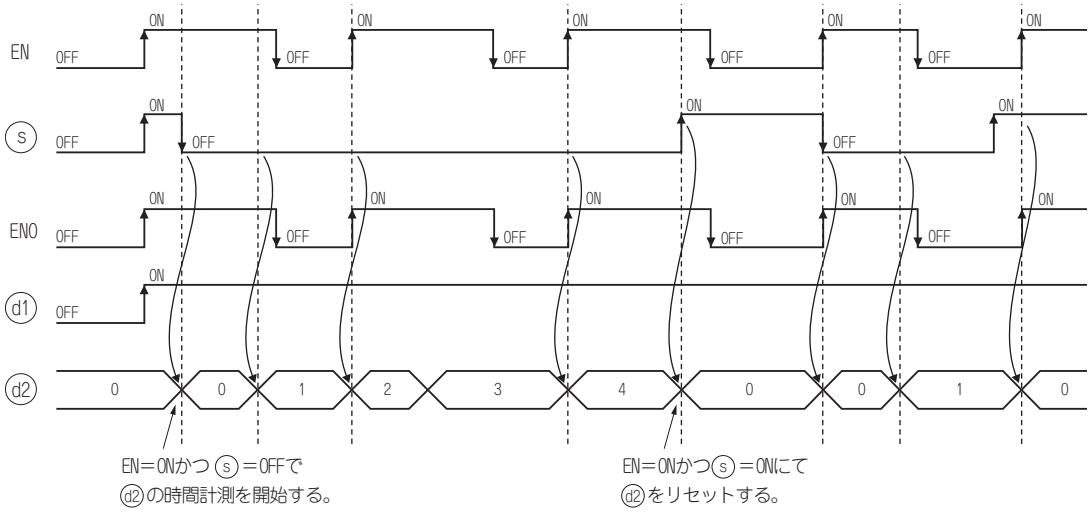
(2) EN/ENO 付き関数

実行条件と演算結果を，下記に示します。

EN	ENO	ⓓ1，ⓓ2
TRUE（演算実行）	TRUE	演算出力値
FALSE（演算停止）	FALSE	前回出力値

[タイミングチャート]

n＝T#5s(5秒)の場合



エラー

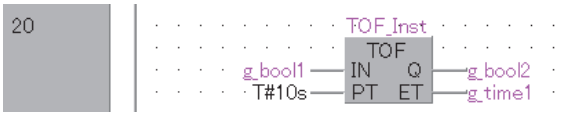
TOF(_E) での演算エラーはありません。

プログラム例

ⓢに入力されたビット型のデータが ON した時にⓓ1のビット型のデータを ON し，ⓢが OFF したとき，10 秒後にⓓ1を OFF するプログラム。

(a) EN/ENO なし関数 (TOF)

〔構造化ラダー/FBD〕

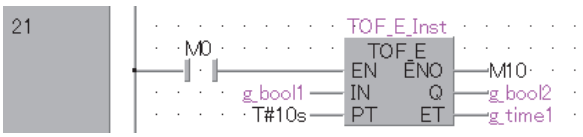


〔ST〕

```
TOF_Inst(IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1 );
```

(b) EN/ENO 付き関数 (TOF_E)

〔構造化ラダー/FBD〕



〔ST〕

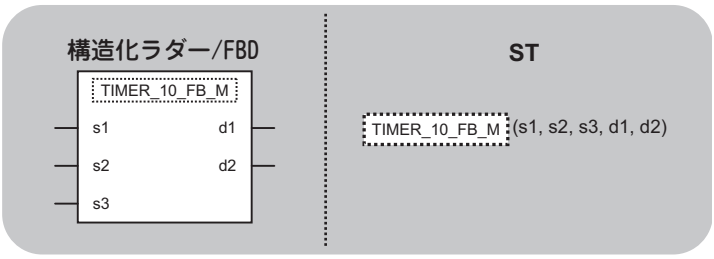
```
TOF_E_Inst(EN:= M0 ,IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1
,ENO:= M10 );
```

5.12.4 タイマファンクションブロック

TIMER_10_FB_M

Basic | High performance | Process | Redundant | Universal | LCPU

TIMER_10_FB_M
TIMER_100_FB_M
TIMER_HIGH_FB_M
TIMER_LOW_FB_M
TIMER_CONT_FB_M
TIMER_CONTHFB_M



TIMER_10_FB_M には，下記の関数が入ります。

TIMER_10_FB_M	TIMER_100_FB_M
TIMER_HIGH_FB_M	TIMER_LOW_FB_M
TIMER_CONT_FB_M	TIMER_CONTHFB_M

入力引数，	s1(Coil):	実行条件（TRUE：実行　FALSE：停止）	:ビット
	s2(Preset):	タイマ設定値	:ワード［符号付き］
	s3(ValueIn):	タイマ初期値	:ワード［符号付き］
出力引数，	d1(ValueOut):	タイマ現在値	: ANY16
	d2(Status):	出力	:ビット

機　能

演算処理

(1) TIMER_10_FB_M

(a) ⓢ1 の実行条件が ON すると，現在値の計測を開始します。
ⓢ3 ×10ms から計測を開始し，ⓢ2 ×10ms まで計測値が到着すると，ⓓ2 が ON します。
現在の計測値は，ⓓ1 に出力されます。

(b) ⓢ1 の実行条件が OFF すると，現在値はⓢ3 の値となり，ⓓ2 も OFF します。

(c) 高速タイマの計測単位（時限設定）を PC パラメータのデフォルト値から変更した場合，コンパイル時に C9047 のワーニングが発生します。

(d) ⓢ2 の有効設定範囲は，0 ～ 32767 です。

(e) ⓢ3 の有効設定範囲は，-32768 ～ 32767 です。ただし，負の値を指定した場合，0 が初期値となります。

(2) TIMER_100_FB_M

(a) ⓢ1 の実行条件が ON すると，現在値の計測を開始します。
ⓢ3 ×100ms から計測を開始し，ⓢ2 ×100ms まで計測値が到達するとⓓ2 が ON します。
現在の計測値をⓓ1 から出力します。

(b) ⓢ1 の実行条件が OFF すると，現在値はⓢ3 の値となり，ⓓ2 も OFF します。

(c) 低速タイマの計測単位（時限設定）を PC パラメータのデフォルト値から変更した場合，コンパイル時に C9214 のワーニングが発生します。

(d) ⓢ2 の有効設定範囲は，0 ～ 32767 です。

(e) ⓢ3 の有効設定範囲は，-32768 ～ 32767 です。ただし，負の値を指定した場合，0 が初期値となります。

(3) TIMER_HIGH_FB_M

(a) 計測単位が 0.1 ～ 100ms の高速タイマです。ⓢ1 の実行条件が ON すると，現在値の計測を開始します。ⓢ3 ×0.1 ～ 100ms から計測を開始し，ⓢ2 ×0.1 ～ 100ms まで計測値が到達するとⓓ2 が ON します。
現在の計測値をⓓ1 から出力します。

(b) ⓢ1 の実行条件が OFF すると，現在値はⓢ3 の値となり，ⓓ2 も OFF します。

(c) 高速タイマの計測単位（時限）は，デフォルト値が 10ms です。
計測単位は下記の範囲で変更できます。
設定は，PC パラメータの PC システム設定で行います。

CPU ユニット	設定単位
ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU	0.1ms ～ 100ms
ユニバーサルモデル QCPU，LCPU	0.01ms ～ 100ms

(d) ⓢ2 の有効設定範囲は，0 ～ 32767 です。

(e) ⓢ3 の有効設定範囲は，-32768 ～ 32767 です。ただし，負の値を指定した場合，0 が初期値となります。

(4) TIMER_LOW_FB_M

(a) 計測単位が 1 ～ 1000ms の低速タイマです。ⓢ1 の実行条件が ON すると，現在値の計測を開始します。ⓢ3 ×1 ～ 1000ms から計測を開始し，ⓢ2 ×1 ～ 1000ms まで計測値が到達するとⓓ2 が ON します。
現在の計測値をⓓ1 から出力します。

(b) ⓢ1 の実行条件が OFF すると，現在値はⓢ3 の値となり，ⓓ2 も OFF します。

(c) 低速タイマの計測単位（時限）は，デフォルト値が 100ms です。
計測単位は 1 ～ 1000ms で，1ms 単位で変更できます。
設定は，PC パラメータの PC システム設定で行います。

(d) ⓢ2の有効設定範囲は，0 ～ 32767 です。

(e) ⓢ3の有効設定範囲は，-32768 ～ 32767 です。ただし，負の値を指定した場合，0 が初期値となります。

(5) TIMER_CONT_FB_M,TIMER_CONTHFB_M

(a) 変数が ON している時間を計測する積算タイマです。ⓢ1の実行条件が ON すると，現在値の計測を開始します。積算タイマには低速積算タイマ TIMER_CONT_FB_M と高速積算タイマ TIMER_CONTHFB_M の２種類があります。
ⓢ3×1 ～ 1000ms から計測を開始し，ⓢ2×1 ～ 1000ms まで計測値が到達するとⓓ2が ON します。現在の計測値をⓓ1から出力します。

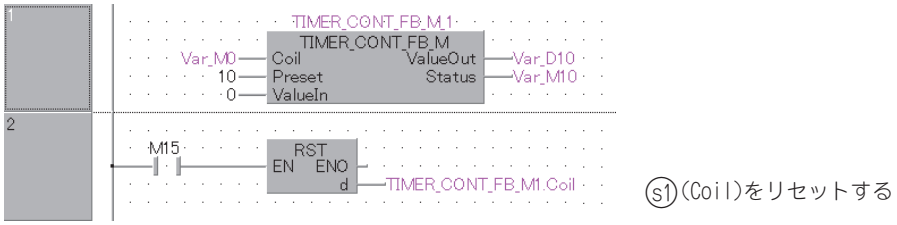
(b) ⓢ1の実行条件が OFF しても，計測値ⓓ1，ⓓ2の ON/OFF 状態を保持します。再度ⓢ1の実行条件が ON すると，保持していた計測値から計測を再開します。

(c) 積算タイマの計測単位（時限）は，低速タイマ TIMER_LOW_FB_M，高速タイマ TIMER_HIGH_FB_M と同一です。
- 低速積算タイマ：低速タイマ
- 高速積算タイマ：高速タイマ

(d) ⓢ2の有効設定範囲は，0 ～ 32767 です。

(e) ⓢ3の有効設定範囲は，-32768 ～ 32767 です。ただし，負の値を指定した場合，0 が初期値となります。

(f) 積算タイマの計測値をリセットしたい場合は，ⓢ1をリセットしてください。
（例）インスタンス名が TIMER_CONT_FB_M_1 の場合
〔構造化ラダー/FBD〕



〔ST〕

```
TIMER_CONT_FB_M_1(Coil:=Var_M0,Preset:=10,ValueIn:=0,ValueOut:=Var_
D10,Status:=Var_M10);
RST(M15,TIMER_CONT_FB_M_1.Coil);
```

エラー

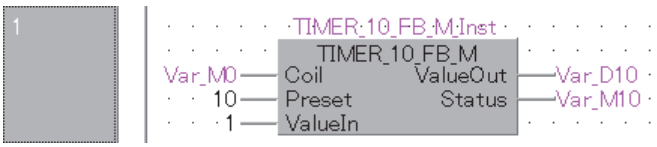
タイマファンクションブロックでの演算エラーはありません。

プログラム例

(1) TIMER_10_FB_M

ⓢ1 に入力された実行条件が ON し，ⓢ3 ×10ms から ⓢ2 ×10ms まで計測値が到達すると ⓓ2 が ON するプログラム。

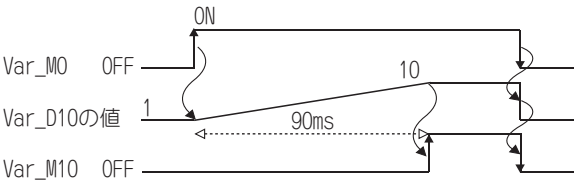
〔構造化ラダー/FBD〕



〔ST〕

```
TIMER_10_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 1 ,ValueOut:=
Var_D10 ,Status:= Var_M10 );
```

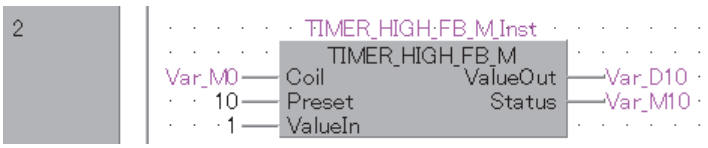
〔タイミングチャート〕



(2) TIMER_HIGH_FB_M

ⓢ1 に入力された実行条件が ON し，ⓢ3 ×10ms から ⓢ2 ×10ms まで計測値が到達すると ⓓ2 が ON するプログラム。

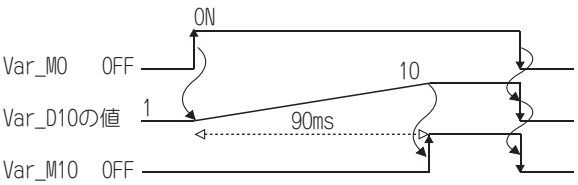
〔構造化ラダー/FBD〕



〔ST〕

```
TIMER_HIGH_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 1 ,ValueOut:=
Var_D10 ,Status:= Var_M10);
```

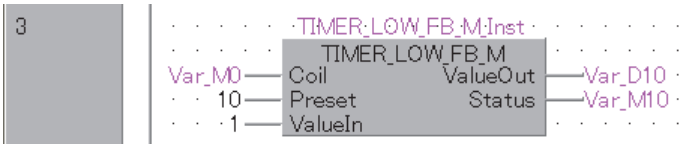
〔タイミングチャート〕



(3) TIMER_LOW_FB_M

ⓢ1 に入力された実行条件が ON し，ⓢ3 ×10ms から ⓢ2 ×100ms まで計測値が到達すると ⓓ2 が ON するプログラム。

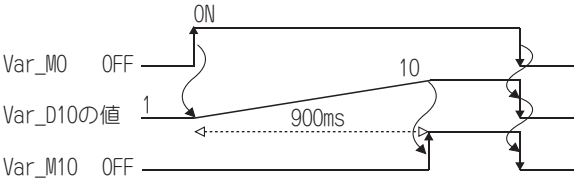
〔構造化ラダー/FBD〕



〔ST〕

```
TIMER_LOW_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 1 ,ValueOut:=
Var_D10 ,Status:= Var_M10);
```

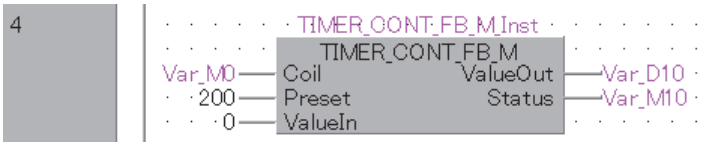
〔タイミングチャート〕



(4) TIMER_CONT_FB_M

ⓢ1 が ON している時間を計測し，ⓢ3 ×10ms から ⓢ2 ×100ms まで計測値が到達すると，ⓓ2 が ON するプログラム。

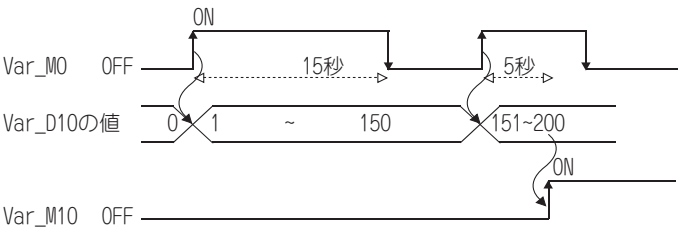
〔構造化ラダー/FBD〕



〔ST〕

```
TIMER_CONT_FB_M_Inst(Coil:= Var_M0 ,Preset:= 200 ,ValueIn:= 0 ,Value
Out:= Var_D10 ,Status:= Var_M10);
```

〔タイミングチャート〕



メ　モ

6 オペレータ

6.1 算術演算 6-2
6.2 ビット型ブール 6-13
6.3 比較演算 6-16

6.1 算術演算

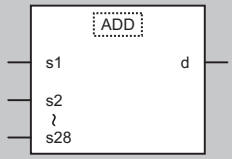
6.1.1 加算

ADD, ＋

Basic | High performance | Process | Redundant | Universal | LCPU

ADD
＋

構造化ラダー/FBD



ST

対応していません。

構造化ラダー/FBD

対応していません。

ST

d:=s1 ＋ s2…s28;

［　　］には，下記のオペレータが入ります。

ADD

［　　］には，下記のオペレータが入ります。

＋

入力引数，	s1 ～ s28:	入力	: ANY_NUM
出力引数，	d:	出力	: ANY_NUM

機　能

演算処理

演算処理については，5.3.1 項を参照してください。

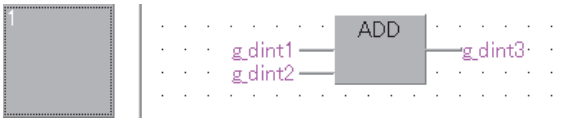
エラー

ADD，＋での演算エラーはありません。

プログラム例

ⓢ1，ⓢ2に入力されたダブルワード［符号付き］型データの加算（ⓢ1＋ⓢ2）を行い，演算結果をⓓからⓢ1，ⓢ2と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint3:= (g_dint1) + (g_dint2);
```

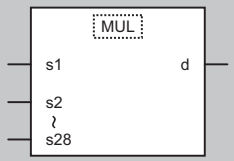
6.1.2 乗算

MUL, ＊

Basic | High performance | Process | Redundant | Universal | LCPU

MUL
＊

構造化ラダー/FBD



ST

対応していません。

構造化ラダー/FBD

対応していません。

ST

d:=s1 ＊ s2…s28;

[ ] には，下記のオペレータが入ります。

MUL

[ ] には，下記のオペレータが入ります。

＊

入力引数，	s1 ～ s28:	入力	: ANY_NUM
出力引数，	d:	出力	: ANY_NUM

機　能

演算処理

演算処理については，5.3.2 項を参照してください。

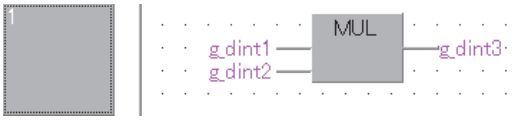
エラー

MUL, ＊での演算エラーはありません。

プログラム例

ⓢ1，ⓢ2に入力されたダブルワード［符号付き］型データの乗算（ⓢ1×ⓢ2）を行い，演算結果をⓓからⓢ1，ⓢ2と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint3:= (g_dint1) * (g_dint2);
```

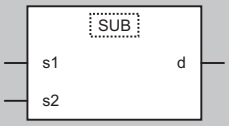
6.1.3 減算

SUB, －

Basic | High performance | Process | Redundant | Universal | LCPU

SUB
－

構造化ラダー/FBD



ST

対応していません。

[ ] には，下記のオペレータが入ります。

SUB

構造化ラダー/FBD

対応していません。

ST

d:=s1 [-] s2;

[ ] には，下記のオペレータが入ります。

－

入力引数，	s1:	入力	: ANY_NUM
	s2:		
出力引数，	d:	出力	: ANY_NUM

機　能

演算処理

演算処理については，5.3.3 項を参照してください。

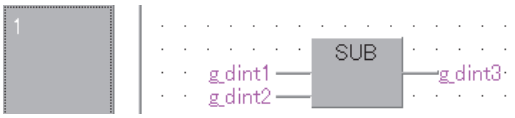
エラー

SUB, －での演算エラーはありません。

プログラム例

ⓢ1, ⓢ2に入力されたダブルワード［符号付き］型データの減算（ⓢ1－ⓢ2）を行い，演算結果をⓓからⓢ1, ⓢ2と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint3:= (g_dint1) - (g_dint2);
```

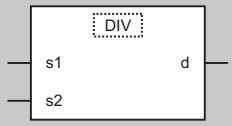
6.1.4 除算

DIV，/

Basic High performance Process Redundant Universal LCPU

DIV
/

構造化ラダー/FBD



ST

対応していません。

[ ] には，下記のオペレータが入ります。

DIV

構造化ラダー/FBD

対応していません。

ST

d:=s1 / s2;

[ ] には，下記のオペレータが入ります。

/

入力引数，	s1:	入力	: ANY_NUM
	s2:		
出力引数，	d:	出力	: ANY_NUM

機　能

演算処理

演算処理については，5.3.4 項を参照してください。

エラー

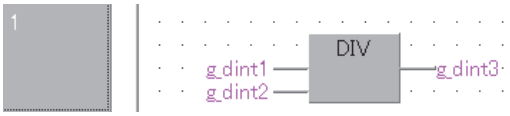
下記の場合には演算エラーとなります。

・ ⓢ2 へ入力される値が0のとき。（0割り）　（エラーコード 4100）

プログラム例

ⓢ1, ⓢ2に入力されたダブルワード [符号付き] 型データの除算（ⓢ1 ÷ ⓢ2）を行い，演算結果の商をⓓからⓢ1, ⓢ2と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_dint3:= (g_dint1) / (g_dint2);
```

6.1.5 剰余

MOD

Basic | High performance | Process | Redundant | Universal | LCPU

MOD

構造化ラダー/FBD	ST
対応していません。	d:=s1 MOD s2;

[ ] には，下記のオペレータが入ります。

MOD

入力引数，	s1: s2:	入力	: ANY_INT
出力引数，	d:	出力	: ANY_INT

機　能

演算処理

演算処理については，5.3.5 項を参照してください。

エラー

下記の場合には演算エラーとなります。

・ ⓢ2 へ入力される値が０のとき。（０割り）　（エラーコード 4100）

プログラム例

ⓢ1，ⓢ2 に入力されたダブルワード [符号付き] 型データの除算

（ⓢ1 ÷ ⓢ2）を行い，演算結果の余りをⓓから ⓢ1，ⓢ2 と同一のデータ型で出力するプログラム。

〔ST〕

```
g_dint3:= (g_dint1) MOD (g_dint2);
```

6.1.6 べき乗

**

Basic | High performance | Process | Redundant | Universal | LCPU

**

構造化ラダー/FBD	ST
対応していません。	d:=s1 ** s2;

[ ] には，下記のオペレータが入ります。

**

入力引数，	s1:	入力	: ANY_REAL
	s2:	入力	: ANY_NUM
出力引数，	d:	出力	: ANY_REAL

機　能

演算処理

演算処理については，5.3.6 項を参照してください。

エラー

本オペレータは，下記の共通命令を使用して作られています。
ⓢ1が単精度実数，ⓢ2がワード［符号付き］の場合：LOG，FLT
ⓢ1が単精度実数，ⓢ2がダブルワード［符号付き］の場合：LOG，DFLT
ⓢ1が単精度実数，ⓢ2が単精度実数 の場合：LOG
ⓢ1が単精度実数，ⓢ2が倍精度実数 の場合：LOGD，DFLTD
ⓢ1が倍精度実数，ⓢ2がワード［符号付き］の場合：LOGD
ⓢ1が倍精度実数，ⓢ2がダブルワード［符号付き］の場合：LOGD，FLTD
ⓢ1が倍精度実数，ⓢ2が単精度実数 の場合：LOGD，DFLTD
ⓢ1が倍精度実数，ⓢ2が倍精度実数 の場合：LOGD

オペレータ実行時に発生するエラーについては，MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編）を参照してください。

プログラム例

べき乗を行い，演算結果をⓓからⓢ1，ⓢ2と同一のデータ型で出力するプログラム。

〔ST〕

```
g_real2:= (g_real1)**(g_int1);
```

6.2 ビット型ブール

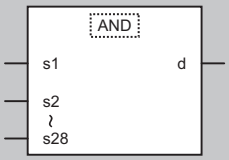
6.2.1 論理積，論理和，排他的論理和，論理否定

AND，&，OR，XOR，NOT

Basic | High performance | Process | Redundant | Universal | LCPU

AND
&
OR
XOR
NOT

構造化ラダー/FBD



ST

対応していません。

構造化ラダー/FBD

対応していません。

ST

d:=s1 AND s2…s28,d);

[ ] には，下記のオペレータが入ります。

AND
OR
XOR

[ ] には，下記のオペレータが入ります。

AND　　&
OR
XOR
NOT

入力引数，	s1 ～ s28: （NOT は s1 のみ ）	入力	: ANY_BIT
出力引数	d:	出力	: ANY_BIT

機　能

演算処理

演算処理については，5.4.1 項を参照してください。

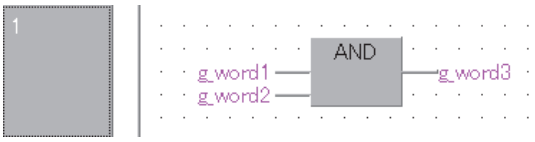
エラー

AND，&，OR，XOR，NOT での演算エラーはありません。

プログラム例

(1) ⓢ1～ⓢ28に入力されたワード [符号なし]/ ビット列 [16 ビット] 型データをビットごとに論理積を行い，演算結果をⓓからⓢ1～ⓢ28と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕

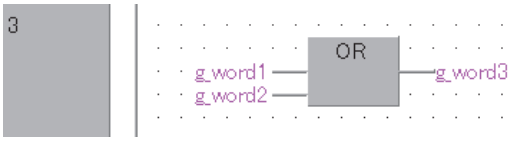


〔ST〕

```
g_word3 :=(g_word1) AND (g_word2);
または
g_word3 :=(g_word1) & (g_word2);
```

(2) ⓢ1～ⓢ28に入力されたワード [符号なし]/ ビット列 [16 ビット] 型データをビットごとに論理和を行い，演算結果をⓓからⓢ1～ⓢ28と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕

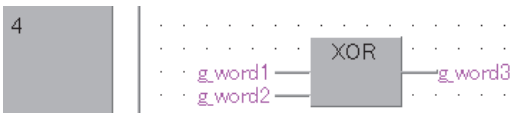


〔ST〕

```
g_word3 :=(g_word1) OR (g_word2);
```

(3) ⓢ1～ⓢ28に入力されたワード [符号なし]/ ビット列 [16 ビット] 型データをビットごとに排他的論理和を行い，演算結果をⓓからⓢ1～ⓢ28と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_word3 :=(g_word1) XOR (g_word2);
```

（4）ⓢ1に入力されたワード［符号なし］/ ビット列［16 ビット］型データをビットごとに論理否定を行い，演算結果をⓓからⓢ1と同一のデータ型で出力するプログラム。

〔構造化ラダー/FBD〕

〔ST〕

```
g_word2 :=NOT (g_word1);
```

6.3 比較演算

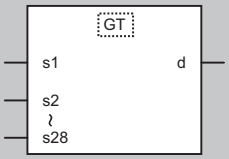
6.3.1 比較

GT，GE，EQ，LE，LT，NE
>，>=，=，<=，<，<>

Basic　High performance　Process　Redundant　Universal　LCPU

GT　>
GE　>=
EQ　=
LE　<=
LT　<
NE　<>

構造化ラダー/FBD



ST

対応していません。

構造化ラダー/FBD

対応していません。

ST

d:=s1 > s2…s28;

[　　] には，下記のオペレータが入ります。

GT
GE
EQ
LE
LT
NE

[　　] には，下記のオペレータが入ります。

>
>=
=
<=
<
<>

入力引数，　s1 ～ s28:　入力　: ANY_SIMPLE
（NE，<> は s1,s2 のみ）
出力引数，　d:　出力（TRUE：真値　FALSE：偽値）　: ビット

機　能

演算処理

演算処理については，5.6.1 項を参照してください。

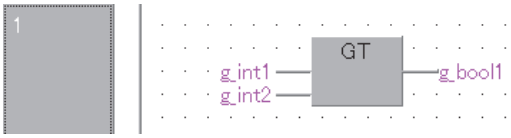
エラー

GT，GE，EQ，LE，LT，NE，>，>=，=，<=，<，<>での演算エラーはありません。

プログラム例

ⓢ1，ⓢ2に入力された入力値の比較演算を行い，演算結果をⓓから出力するプログラム。

〔構造化ラダー/FBD〕



〔ST〕

```
g_bool1 := (g_int1) > (g_int2);
```

メ　モ

索　引

【A】

ABS(_E) 5-127
ADD 6-2
ADD_E 5-131
ADD_TIME(_E) 5-186
AND 6-13
AND_E 5-152

【B】

BCD_TO_DINT(_E) 5-96
BCD_TO_INT(_E) 5-96
BCD_TO_STR(_E) 5-100
BITARR_TO_DINT(_E) 5-113
BITARR_TO_INT(_E) 5-113
BOOL_TO_DINT(_E) 5-2
BOOL_TO_DWORD(_E) 5-7
BOOL_TO_INT(_E) 5-2
BOOL_TO_STR(_E) 5-5
BOOL_TO_TIME(_E) 5-10
BOOL_TO_WORD(_E) 5-7

【C】

CONCAT(_E) 5-175
COUNTER_FB_M 5-218
CPY_BITARR(_E) 5-117
CPY_BIT_OF_INT(_E) 5-123
CTD(_E) 5-211
CTUD(_E) 5-214
CTU(_E) 5-208

【D】

DELETE(_E) 5-180
DINT_TO_BCD(_E) 5-35
DINT_TO_BITARR(_E) 5-115
DINT_TO_BOOL(_E) 5-16
DINT_TO_DWORD(_E) 5-32
DINT_TO_INT(_E) 5-14
DINT_TO_LREAL(_E) 5-22
DINT_TO_REAL(_E) 5-19
DINT_TO_STR(_E) 5-25
DINT_TO_TIME(_E) 5-38
DINT_TO_WORD(_E) 5-29
DIV 6-8
DIV_E 5-140
DIV_TIME(_E) 5-195
DIV(_E) 6-8
DWORD_TO_BOOL(_E) 5-55
DWORD_TO_DINT(_E) 5-61
DWORD_TO_INT(_E) 5-61
DWORD_TO_STR(_E) 5-69
DWORD_TO_TIME(_E) 5-72
DWORD_TO_WORD(_E) 5-66

【E】

EQ 6-16
EQ_E 5-169
EXPT(_E) 5-146

【F】

F_TRIG(_E) 5-206

【G】

GE 6-16
GE_E 5-169
GET_BIT_OF_INT(_E) 5-119
GET_BOOL_ADDR 5-125
GET_INT_ADDR 5-125
GET_WORD_ADDR 5-125
GT 6-16
GT_E 5-169

【I】

INSERT(_E) 5-177
INT_TO_BCD(_E) 5-35
INT_TO_BITARR(_E) 5-115
INT_TO_BOOL(_E) 5-16
INT_TO_DINT(_E) 5-12
INT_TO_DWORD(_E) 5-32
INT_TO_LREAL(_E) 5-22
INT_TO_REAL(_E) 5-19
INT_TO_STR(_E) 5-25
INT_TO_TIME(_E) 5-38
INT_TO_WORD(_E) 5-29

【L】

LE 6-16
LE_E 5-169
LIMITATION(_E) 5-163
LREAL_TO_DINT(_E) 5-44
LREAL_TO_INT(_E) 5-44
LREAL_TO_REAL(_E) 5-49
LT 6-16
LT_E 5-169

【M】

MAXIMUM(_E) 5-160
MID(_E) 5-172
MINIMUM(_E) 5-160
MOD 6-10
MOD_E 5-143
MOVE(_E) 5-149
MUL_E 5-134, 6-4
MUL_TIME(_E) 5-192
MUX(_E) 5-166

【N】

NE 6-16
NE_E 5-169
NOT 6-13
NOT_E 5-152

【O】

OR 6-13
OR_E 5-152

【R】

REAL_TO_DINT(_E) 5-41
REAL_TO_INT(_E) 5-41
REAL_TO_LREAL(_E) 5-47
REAL_TO_STR(_E) 5-51
REPLACE(_E) 5-183

RS(_E) 5-200
R_TRIG(_E) 5-203

【S】

SEL(_E) 5-157
SET_BIT_OF_INT(_E) 5-121
SR(_E) 5-197
STR_TO_BCD(_E) 5-91
STR_TO_BOOL(_E) 5-75
STR_TO_DINT(_E) 5-77
STR_TO_DWORD(_E) 5-84
STR_TO_INT(_E) 5-77
STR_TO_REAL(_E) 5-80
STR_TO_TIME(_E) 5-88
STR_TO_WORD(_E) 5-84
SUB 6-6
SUB_E 5-137
SUB_TIME(_E) 5-189

【T】

TIME_TO_BOOL(_E) 5-103
TIME_TO_DINT(_E) 5-105
TIME_TO_DWORD(_E) 5-110
TIME_TO_INT(_E) 5-105
TIME_TO_STR(_E) 5-108
TIME_TO_WORD(_E) 5-110
TOF_HIGH(_E) 5-226
TOF(_E) 5-226
TON_HIGH(_E) 5-223
TON(_E) 5-223
TP_HIGH(_E) 5-220
TP(_E) 5-220

【W】

WORD_TO_BOOL(_E) 5-55
WORD_TO_DINT(_E) 5-58
WORD_TO_DWORD(_E) 5-64
WORD_TO_INT(_E) 5-58
WORD_TO_STR(_E) 5-69
WORD_TO_TIME(_E) 5-72

【X】

XOR 6-13
XOR_E 5-152

【記号】

− 6-6
< 6-16
& 6-13
* 6-4
** 6-11
+ 6-2
<= 6-16
<> 6-16
/ 6-8
= 6-16
> 6-16
>= 6-16

【エ】

エッジ検出ファンクションブロック一覧 2-7

【オ】

オペレーター一覧 2-8

【カ】

カウンタファンクションブロック一覧 2-7
型変換関数一覧 2-3
関数一覧 2-1
関数一覧表の見方 2-2
関数の構成 3-2
関数の見方 4-1

【サ】

算術演算関数一覧 2-5, 2-8

【シ】

時刻データ型関数一覧 2-6

【セ】

選択関数一覧 2-5

【タ】

タイマファンクションブロック一覧 2-7
単数値変数関数一覧 2-5

【ニ】

二安定ファンクションブロック一覧 2-6
入力ピン数可変関数 3-3

【ヒ】

比較関数一覧 2-6, 2-8
ビット型ブール関数一覧 2-5, 2-8

【モ】

文字列関数一覧 2-6

メ　モ

保証について

ご使用に際しましては，以下の製品保証内容をご確認いただきますよう，よろしくお願いいたします。

１．無償保証期間と無償保証範囲

無償保証期間中に，製品に当社側の責任による故障や瑕疵（以下併せて「故障」と呼びます）が発生した場合，当社はお買い上げいただきました販売店または当社サービス会社を通じて，無償で製品を修理させていただきます。ただし，国内および海外における出張修理が必要な場合は，技術者派遣に要する実費を申し受けます。
また，故障ユニットの取替えに伴う現地再調整・試運転は当社責務外とさせていただきます。

【無償保証期間】

製品の無償保証期間は，お客様にてご購入後またはご指定場所に納入後 36ヵ月とさせていただきます。
ただし，当社製品出荷後の流通期間を最長 6ヵ月として，製造から 42ヵ月を無償保証期間の上限とさせていただきます。また，修理品の無償保証期間は，修理前の無償保証期間を超えて長くなることはありません。

【無償保証範囲】

(1) 一次故障診断は，原則として貴社にて実施をお願い致します。
ただし，貴社要請により当社，または当社サービス網がこの業務を有償にて代行することができます。この場合，故障原因が当社側にある場合は無償と致します。

(2) 使用状態・使用方法，および使用環境などが，取扱説明書，ユーザーズマニュアル，製品本体注意ラベルなどに記載された条件・注意事項などに従った正常な状態で使用されている場合に限定させていただきます。

(3) 無償保証期間内であっても，以下の場合には有償修理とさせていただきます。
①お客様における不適切な保管や取扱い，不注意，過失などにより生じた故障およびお客様のハードウェアまたはソフトウェア設計内容に起因した故障。
②お客様にて当社の了解なく製品に改造などの手を加えたことに起因する故障。
③当社製品がお客様の機器に組み込まれて使用された場合，お客様の機器が受けている法的規制による安全装置または業界の通念上備えられているべきと判断される機能・構造などを備えていれば回避できたと認められる故障。
④取扱説明書などに指定された消耗部品が正常に保守・交換されていれば防げたと認められる故障。
⑤消耗部品（バッテリ，リレー，ヒューズなど）の交換。
⑥火災，異常電圧などの不可抗力による外部要因および地震，雷，風水害などの天変地異による故障。
⑦当社出荷当時の科学技術の水準では予見できなかった事由による故障。
⑧その他，当社の責任外の場合またはお客様が当社責任外と認めた故障。

２．生産中止後の有償修理期間

(1) 当社が有償にて製品修理を受け付けることができる期間は，その製品の生産中止後 7 年間です。
生産中止に関しましては，当社テクニカルニュースなどにて報じさせていただきます。

(2) 生産中止後の製品供給（補用品も含む）はできません。

３．海外でのサービス

海外においては，当社の各地域 FA センターで修理受付をさせていただきます。ただし，各 FA センターでの修理条件などが異なる場合がありますのでご了承ください。

４．機会損失，二次損失などへの保証責務の除外

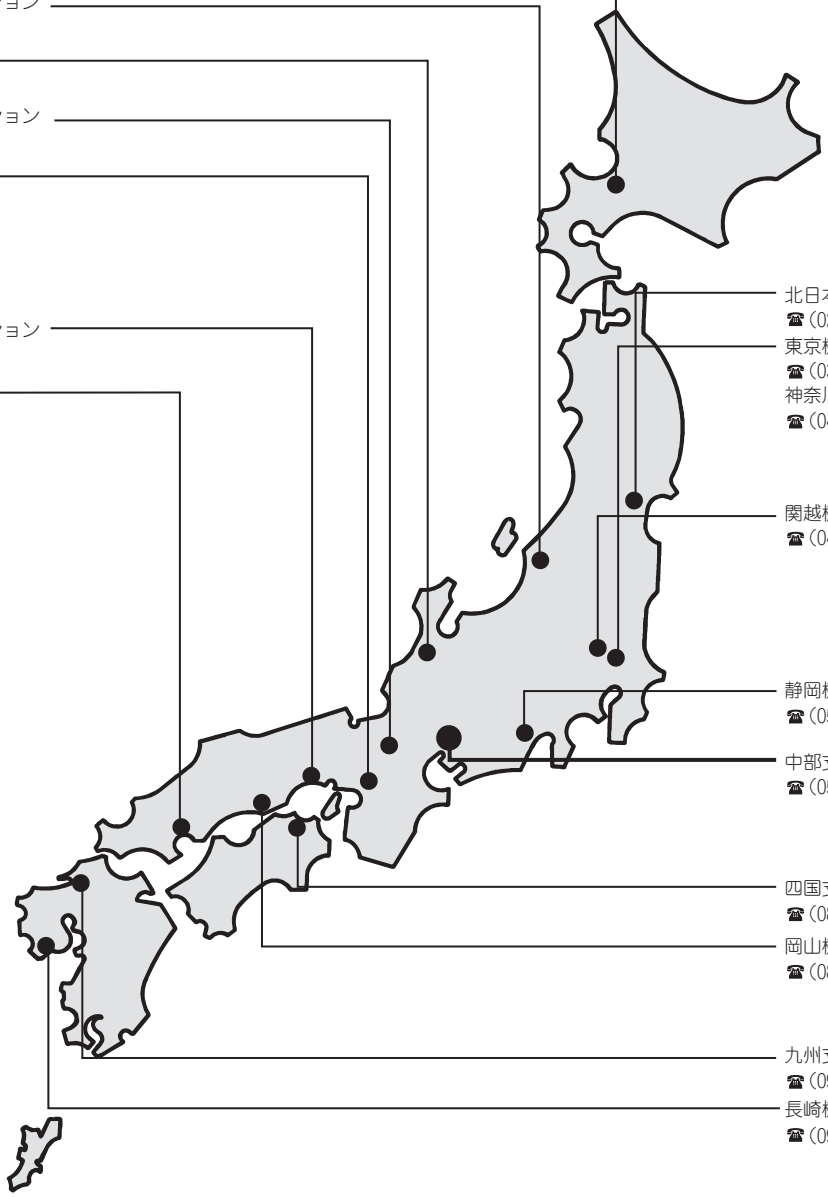
無償保証期間の内外を問わず，当社の責に帰すことができない事由から生じた障害，当社製品の故障に起因するお客様での機会損失，逸失利益，当社の予見の有無を問わず特別の事情から生じた損害，二次損害，事故補償，当社製品以外への損傷，およびお客様による交換作業，現地機械設備の再調整，立上げ試運転その他の業務に対する補償については，当社責務外とさせていただきます。

５．製品仕様の変更

カタログ，マニュアルもしくは技術資料などに記載の仕様は，お断りなしに変更させていただく場合がありますので，あらかじめご承知おきください。

以　上

サービスネットワーク（三菱電機システムサービス株式会社）



Microsoft，Windows，Windows NT，Windows Vista は，米国 Microsoft Corporation の米国およびその他の国における登録商標です。
Pentium は，Intel Corporation の米国およびその他の国における商標です。
Ethernet は，米国 Xerox Corporation の商標です。
PC-9800，PC98-NX は，日本電気株式会社の登録商標です。
SD ロゴ，SDHC ロゴは商標です。
その他，本文中における会社名，商品名は，各社の商標または登録商標です。

SH(名)-080737-0

MELSEC-Q/L
構造化プログラミングマニュアル
応用関数編

三菱電機株式会社 〒100-8310 東京都千代田区丸の内2-7-3(東京ビル)

お問い合わせは下記へどうぞ

本社機器営業部 …… 〒100-8310 東京都千代田区丸の内2-7-3(東京ビル) …… (03)3218-6760
北海道支社 …… 〒060-8693 札幌市中央区北二条西4-1(北海道ビル) …… (011)212-3794
東北支社 …… 〒980-0011 仙台市青葉区上杉1-17-7(仙台上杉ビル) …… (022)216-4546
関越支社 …… 〒330-6034 さいたま市中央区新都心11-2(明治安田生命さいたま新都心ビル) …… (048)600-5835
新潟支店 …… 〒950-8504 新潟市中央区東大通2-4-10(日本生命ビル) …… (025)241-7227
神奈川支社 …… 〒220-8118 横浜市西区みなとみらい2-2-1(横浜ランドマークタワー) …… (045)224-2624
北陸支社 …… 〒920-0031 金沢市広岡3-1-1(金沢パークビル) …… (076)233-5502
中部支社 …… 〒451-8522 名古屋市西区牛島町6-1(名古屋ルーセントタワー) …… (052)565-3314
豊田支店 …… 〒471-0034 豊田市小坂本町1-5-10(矢作豊田ビル) …… (0565)34-4112
関西支社 …… 〒530-8206 大阪市北区堂島2-2-2(近鉄堂島ビル) …… (06)6347-2771
中国支社 …… 〒730-8657 広島市中区中町7-32(ニッセイ広島ビル) …… (082)248-5348
四国支社 …… 〒760-8654 高松市寿町1-1-8(日本生命高松駅前ビル) …… (087)825-0055
九州支社 …… 〒810-8686 福岡市中央区天神2-12-1(天神ビル) …… (092)721-2247

三菱 FA 検索

www.MitsubishiElectric.co.jp/fa/

メンバー登録無料!

インターネットによる情報サービス「三菱電機FAサイト」

三菱電機FAサイトでは、製品や事例などの技術情報に加え、トレーニングスクール情報や各種お問い合わせ窓口をご提供しています。また、メンバー登録いただくとマニュアルやCADデータ等のダウンロード、eラーニングなどの各種サービスをご利用いただけます。

三菱電機FA機器電話, FAX技術相談

●電話技術相談窓口 受付時間※1 月曜～金曜 9:00～19:00、土曜・日曜・祝日 9:00～17:00

対象機種			電話番号
シーケンサ	MELSEC-Q/L/QnA/Aシーケンサー般(下記以外)		052-711-5111
	MELSEC-F FX/Fシーケンサ全般		052-725-2271※2
	ネットワークユニット/シリアルコミュニケーションユニット		052-712-2578
	アナログユニット/温調ユニット/温度入力ユニット/高速カウンタユニット		052-712-2579
	MELSOFT シーケンサプログラミングツール	MELSOFT GXシリーズ SW□IVD-GPPA/GPPQなど	052-711-0037
	MELSOFT 統合エンジニアリング環境	MELSOFT iQ Works(Navigator)	052-712-2370
	MELSOFT 通信支援ソフトウェアツール	MELSOFT MXシリーズ SW□D5F-CSKP/OLEX/XMOPなど	
	MELSECパソコンボード	Q80BDシリーズなど	
	C言語コントローラ/MESインタフェースユニット/高速データロガーユニット		
	iQ Sensor Solution		
	MELSEC計装/Q二重化	プロセスCPU 二重化CPU MELSOFT PXシリーズ	052-712-2830※2
	MELSEC Safety	安全シーケンサ(MELSEC-QSシリーズ) 安全コントローラ(MELSEC-WSシリーズ)	052-712-3079※2
	電力計測/絶縁監視ユニット	QE8□シリーズ	052-719-4557※2※3
表示器		GOT-F900/DUシリーズ	052-725-2271※2
		GOT1000/A900シリーズなど MELSOFT GTシリーズ	052-712-2417
サーボ/位置決めユニット/モーションコントローラ		MELSERVOシリーズ 位置決めユニット/シンプルモーションユニット モーションCPU(Q/Aシリーズ) MELSOFT MTシリーズ/MRシリーズ	052-712-6607
インバータ		FREQROLシリーズ	052-722-2182
ロボット		MELFAシリーズ	052-721-0100

※1:春季・夏季・年末年始の休日を除く ※2:金曜は17:00まで ※3:土曜・日曜・祝日を除く

●FAX技術相談窓口 受付時間※4 9:00～16:00(受信は常時※5)

対象機種	FAX番号
上記電話技術相談対象機種	052-719-6762
電力計測/絶縁監視ユニット(QE8□シリーズ)	084-926-8340

三菱電機FAサイトの「仕様・機能に関するお問い合わせ」もご利用ください。
※4:土曜・日曜・祝日、春季・夏季・年末年始の休日を除く ※5:春季・夏季・年末年始の休日を除く

本マニュアルは, 輸出する場合, 経済産業省への役務取引許可申請は不要です。

形名	Q-KP-OK-J
形名コード	13JC19
SH(名)-080737-O(1304)KWIX	

この印刷物は2013年4月の発行です。なお, お断りなしに仕様を変更することがありますのでご了承ください。 2013年4月作成

この標準価格には消費税は含まれておりません。ご購入の際には消費税が付加されますのでご了承おき願います。 標準価格 3,000円